Universidad
Carlos III de Madrid

Departamento de Tecnología Electrónica

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

Plataforma de riego controlado basada en microcontrolador de bajo coste y amplia conectividad

Autor: Nerea Rodera Sánchez

Tutor: Pablo Zumel Vaquero

Leganés, Septiembre 2017

Agradecimientos

En primer lugar, quiero agradecer a mi familia, a mis padres y a mi hermano Iván, su apoyo incondicional y su fe en mí y en mi trabajo, incluso cuando ni siquiera yo lo tenía claro. Sin ellos nunca lo hubiera conseguido.

También a Jesús P., Agustín, Jesús A., César y Aitor, de los laboratorios, por su amabilidad, su alegría y su inestimable ayuda en todo lo referido a la programación; y porque el tiempo trabajando con ellos ha sido lo mejor de mi paso por la universidad. Igualmente, gracias a los chicos del máster: a Dragos, a Carlos y a Manuel, por dedicarme su tiempo, aun estando muy ocupados. Su aportación al trabajo y sus consejos no son cuantificables.

Además, quiero agradecerle a mis compañeros de clase el haber podido hacer esto todos juntos. En especial agradecerle a Luis, su infinita paciencia, su apoyo y sus grandes ideas, entre otras muchas cosas; y a Víctor, el ser el mejor compañero que uno puede desear en clase. También a los que ya no estudian conmigo, pero que se han esforzado por estar ahí. Gracias César, Jose, Dani y Elena.

Asimismo, quiero darle las gracias a mi tutor, Pablo Zumel, por aceptar llevarme este trabajo y ayudarme a darle forma a una idea hasta convertirla en lo que es hoy.

Por último, a ti, Rober. Allá donde estés, sé que estarás orgulloso de que he aprendido a programar casi tan bien como tú. Este trabajo también es tuyo.

Resumen

El riego automatizado es una realidad hoy día. En el proyecto que nos ocupa, se verá un modelo de riego automatizado orientado a climas continentales —cambios bruscos de temperatura y humedad baja— que tiene en cuenta la temperatura, la humedad, la fecha del último riego y las capacidades hídricas de cada planta, así como el tipo de terreno en el que esta se encuentre. Para la elección del terreno y el tiempo de planta, se hará uso una interfaz web, a la que se podrá acceder por WiFi, contando con dos terrenos disponibles y un total de seis ambientes de exterior y tres de interior. Mientras, la temperatura y la humedad serán medidas por la tarjeta Sensor Hub BoosterPack de Tiva que irá acoplado al microcontrolador EK-TM4C1924XL, el cual, a su vez, permitirá llevar a cabo el programa de riego.

Una vez seleccionados los parámetros, el programa se pondrá en funcionamiento para regar pasado un número determinado de horas, siempre que se cumplan las condiciones climatológicas adecuadas. Así, el riego se realizará siempre que se encuentre en una temperatura comprendida entre los 0 y los 40 grados centígrados, y con una humedad inferior al 50 %. De este modo, se evitará regar cuando haya riesgos de congelación, o cuando haga tanto calor que sea perjudicial para las plantas. Del mismo modo, controlando el valor de la humedad se asegurará de que sólo se riega cuando la planta lo necesita, consiguiendo así un mayor ahorro energético.

Debido a su flexibilidad respecto a los ambientes —permitiendo cambiar estos en cualquier momento— y a su bajo coste, este sistema es altamente adecuado tanto para fines comerciales (invernaderos, superficies comerciales como viveros) como para particulares, dado que no necesita mantenimiento y es fácilmente adaptable a sistemas de riego no automatizados previos. Por otro lado, en caso de no tener un sistema de riego previo, al final de la memoria se sugiere uno, aunque el usuario deberá tener en cuenta sus propias necesidades.

Por último, se debe añadir que, teniendo en cuenta las extensiones de cada riego o su posible ubicación, se han implementado dos tipos de riego diferentes. Uno de ellos, para el césped, mediante aspersores; el otro, para el resto de ambientes, se efectuará mediante riego por goteo.

Palabras clave: riegos, automatización, web.

Abstract

Nowadays, automatic irrigation is a reality. At the present project, we will see an automatic irrigation model oriented towards a Continental weather — abrupt temperature changes and low humidity values— which considers location's temperature and humidity, last irrigation date, water capacities of each plant and the kind of ground in which we will work. For ground and plants choices we will use a web interface, to which we could access by WiFi, in which we have two different kinds of ground (arid and wet) and nine types of ambiences (six of exterior and three of interior plants). Meantime, temperature and humidity will be measured by Tiva's Sensor Hub BoosterPack, that is coupled to EK-TM4C1924XL microcontroler. This one, on the other hand, will be in charge of irrigation program.

Once all parameters are selected, program will start, making an irrigation once that appropriate time has passed and if climatologic conditions are correct. So, irrigation will be a fact if temperature is between 0 and 40 degrees and humidity is lower than 50 %. This way, we will avoid to irrigating when there is risk of freezing or when the whether is so hot that could damages the plants. In the same way, if we control the humidity value, we will assure that the program only irrigates when plant needs it, so we can get a greater energy saving.

Due to the program flexibility about the ambiences —allowing the user to change the ambiences in any moment— and the low cost, this system is highly appropriated both for commercial purposes (greenhouses, commercial areas) and for private gardens, because the system does not need maintenance and it is easily adaptable to previous non-automated irrigation systems. On the other hand, in case of not having one, at the end of this document there is one suggested, although the users should have to take in count their own needs.

At last, it is necesary to add that, having present the irrigation area or this location, we have been implemented two different kinds of irrigation. One of them, for grass, by sprinklers; the other one, for the rest of the ambiences, by drip irrigation.

Keywords: automatic, irrigation, web.

Índice

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

1.1. Breve descripción

En este proyecto se ha desarrollado un sistema de riego automatizado basado en el microcontrolador de bajo coste EK-TM4C1294XL de Tiva y en su complemento Sensor Hub BoosterPack. Se tratará de un sistema que tenga en cuenta las necesidades hídricas —tiempo de riego y tiempo entre riegos— de cada planta, así como la temperatura, la humedad y el tipo de terreno. Para la elección del tipo de planta que se desea, así como del tipo de terreno sobre el que se encuentran, se contará con una web alojada en el microprocesador. También, gracias a esta web, se podrán monitorizar la temperatura, humedad y temperatura aparente, así como el estado de riego en cada instante. Se podrá acceder a estos recursos tanto desde un ordenador como desde un sistema android a través de una red WiFi.

Como se muestra en la figura 1.1, el microprocesador (junto con la tarjeta de sensores, que irá conectada a los pines de este), estará conectado a 5 voltios[1] y tendrá disponibles dos pines de salida, el PL0 y el PL1, que se activarán en caso de riego según el ambiente elegido. Así, si se ha elegido césped, se activará el pin 1, que estará conectado a unos aspersores; y, en cambio, si se ha elegido cualquiera de los otros ambientes, el riego será por goteo, activando el pin 0. Se ha elegido este método de diferenciación con el objetivo de ahorrar la mayor cantidad posible de agua teniendo en cuenta, también, las necesidades de cada planta.

[1] También puede ir conectado a 3,3 voltios.

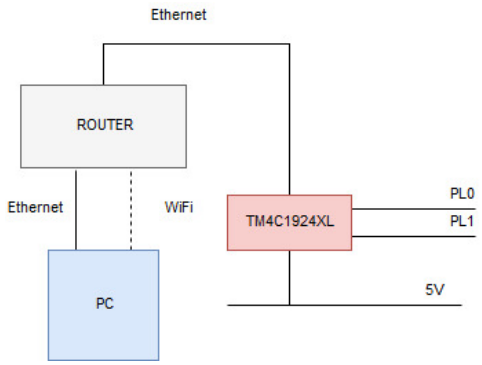


Figura 1.1: Esquema de conexión de los elementos

Por otro lado, el microcontrolador también irá conectado mediante un cable de red al router[2], que a su vez se conectará al ordenador mediante otro cable o vía WiFi, en función del router y las necesidades del cliente (si se elige la opción WiFi se podrá conectar con un smartphone o tableta). De esta manera, se podrá acceder a la web donde registrarse y elegir el ambiente y el terreno o directamente acceder para ver el estado actual. También, mediante el formulario de registro, se podrá cambiar de ambiente tantas veces como sea necesario, puesto que no pondrá problema si existe ese nombre de usuario previamente.

Además de la web, se han utilizado programas de ejemplo para medir la temperatura y la humedad y para obtener la dirección IP con la que se podrá acceder a la web desde el navegador. También, se ha creado un programa de riego que tiene en cuenta los parámetros elegidos y los tiempos de riego, así como las condiciones climáticas, para regar sólo cuando las temperaturas comprenden entre 0 y 40 grados y la humedad es menor del 50 %, adaptándose así a la climatología usual de un clima continental[3]. En caso de no cumplir las condiciones, se mostraría en la web, junto con el tiempo restante al próximo riego.

Por último, se debe añadir que, para simular de todo el proceso, no sólo se ha creado la interfaz web y el programa de riego, si no que se ha buscado un circuito de riego que comprende desde la fuente de alimentación del microcontrolador hasta los aspersores o cabezales de riego, pasando por la bomba o las electroválvulas que controlarán el riego. Este se explica en el capítulo 7, pero siempre como sugerencia, puesto que cada elemento dependerá del uso que se le quiera dar. También se apunta que, en caso de tener un sistema de riego previo, este es fácilmente adaptable a la automatización con el microprocesador a través de una electroválvula.

1.2. Objetivos del proyecto

Este proyecto nació con el objetivo de hacer más fácil la tarea de regadío, a nivel doméstico o profesional, de manera sencilla y económica. Principalmente, se debía informar en todo momento al usuario de las condiciones climáticas del lugar y, si estas eran

[2]Esto se podría hacer vía WiFi con un módulo anexo al microcontrolador.
[3]Para otro tipo de clima más húmedo habría que reajustar este parámetro.

las idóneas, efectuar el riego.

Sin embargo, esta idea no tenía en cuenta las necesidades hídricas de cada tipo de planta. Esto podía llegar a suponer desde un derroche innecesario de agua, en caso de que el usuario requiriese un tipo de planta con menor necesidad de la programada, hasta la muerte de la planta por falta de riego, en la situación contraria.

De este modo, la idea original se mantuvo, pero ahora los objetivos del trabajo se centran, en mayor grado, en satisfacer las necesidades de la planta y del medioambiente. Así pues, enumerándolos de manera breve, estos serían:

- Sensorización de magnitudes ambientales y control del riego.
- Selección del tipo de planta y terreno que se pretende regar.
- Bajo coste.
- Alta conectividad para interactuar con el sistema a distancia y desde un terminal movil.

Por esta razón se ha elegido el microcontrolador EK-TM4C1294XL, al ser un producto de coste bajo y, a la vez, ofrecer una gran versatilidad para la implementación del proyecto, como su amplia conectividad o su posibilidad de alojar una página web gracias a su sistema operativo.

1.3. Estructura del documento

A continuación, se hará un breve resumen del contenido de cada capítulo a fin de facilitar la lectura del documento.

- En el capítulo 1 se realiza una introducción, con una breve descripción del trabajo y los objetivos que persigue.
- En el capítulo 2 se habla de los sistemas de riego, de los proyectos similares y del marco regulatorio y el entorno socio-económico en el que se encuentra este proyecto.
- El capítulo 3 se ha dedicado a exponer las herramientas hardware y software que se ha utilizado, haciendo un recorrido por los entornos de desarrollo y lenguajes de programación.
- En el capítulo 4 se explica con detalle cómo son las páginas web que contiene el microprocesador.
- Los capítulos 5 y 6 se han utilizado para detallar el programa de riego. Más concretamente, en el capítulo 5 se habla del programa base, *enet_io*; mientras, en el 6, se comentan las modificaciones hechas a este para conseguir un programa de riego.
- El último capítulo, el 7, está dedicado al sistema de riego externo al microcontrolador, como válvulas, bombas, fuentes de alimentación... etc.

Tras estos capítulos, se pueden ver cuatro anexos, que son:

- Anexo 1: Tablas con los tiempos de riego asumidos para cada tipo de ambiente y terreno.
- Anexo 2: Presupuesto detallado del coste del proyecto, así como un presupuesto supuesto del coste de una producción por lotes y su inserción en el marcado.
- Anexo 3: Código HTML completo de las páginas web.
- Anexo 4: Código C del programa de riego incluido en el programa *enet_io*.

Capítulo 2

Análisis del estado del arte

2.1. Introducción a los sistemas de riego

> «*Los sistemas de riego han recorrido un largo camino desde las antiguas tribus nómadas hasta hoy en día. Sin embargo, siempre han supuesto un factor crucial en el desarrollo y crecimiento de la industria agrícola*»
>
> (DIY START, 2015: [5])

Al principio, nuestros antepasados se dedicaban a la caza y a la pesca. Sin embargo, con la revolución neolítica, llegó también el cultivo de algunos cereales como el trigo o el arroz. [6] Esto fue, aproximadamente, en el 6000 a.C.

Desde entonces, el ser humano ha avanzado notablemente en el desarrollo de nuevas técnicas y herramientas para facilitar las tareas de riego. La cultura egipcia, alrededor del año 3500 a.C. comenzó a usar el «nilómetro» una columna vertical sumergida en el río Nilo con la que pretendían medir las crecidas y aprovecharlas para el riego.[45]

Además de este método, otras culturas como la Maya o la Azteca utilizaban canales o campos en elevados fácilmente inundables o terrenos en desnivel. Los árabes, por su parte, idearon un sistema de canales con ramificaciones para el regadío llamados acequias.

Sin embargo, estos son métodos que aprovechaban los recursos naturales de forma pasiva. En el siglo I a.C. el Imperio Romano empezó a construir los famosos acueductos, unas construcciones de puentes con canales internos que servía para abastecer de agua las ciudades y los campos de riego.

Pero no fue hasta 1673 en que se utilizó la primera manguera, de cuero. La de goma llegó en 1835. [31] Desde entonces, la manguera se convirtió, durante muchos años, en uno de los métodos de riego más usado, tanto a nivel particular como en grandes cultivos, combinado con acequias. También se puede nombrar la regadera como herramienta de riego a nivel de pequeños cultivos.

Ahora bien, las llegadas del aspersor (1872) y, posteriormente, el riego por goteo (1964), han supuesto un antes y un después en el regadío tal y como se conocía; y la automatización de tales procesos a llevado a un ahorro de tiempo y recursos que nunca antes se había creído posible.

En la figura 2.1 se puede ver uno de los primeros modelos de aspersor, el aspersor de mariposa, que fue fabricado en 1891 y muy utilizado en épocas posteriores.[32]

Figura 2.1: Aspersor de mariposa ©Riegos Grandal. Imagen obtenida de: [32]

2.2. Estado de la técnica

Se ha visto que los sistemas de riego han recorrido un largo camino desde la prehistoria hasta el momento actual. El aspersor supuso una innovación a nivel agrícola pero, aún así, requería de una o varias personas pendientes de las condiciones climatológicas para poner en marcha o parar el sistema. Gracias a la automatización de procesos y a la llegada de los riegos inteligentes, se ha facilitado la tarea del regadío de grandes superficies de cultivo o invernaderos. Además, estos sistemas permiten un ahorro en el consumo de agua y una mayor eficiencia a la hora del cuidado de las plantas.

A continuación, se mostrarán algunos ejemplos de los riegos inteligentes más avanzados del mercado con algunas características similares al propuesto en este proyecto como son el software libre, la monitorización de temperatura y humedad o la interfaz web accesible desde diferentes dispositivos:

Iberdrola

La compañía de gas y electricidad Iberdrola ofrece un riego inteligente muy similar al de este proyecto. Se trata de un sistema orientado a jardines particulares y programable desde un smartphone o tablet con conexión a internet y, además, autoajusta el riego a partir de la previsión del tiempo, evita las heladas y da informes personalizados para un mayor ahorro en el consumo de agua. [14]

Este modelo de riego contiene un caudalímetro y un sensor de lluvia, y puede operar entre los -20 y los 55 grados centígrados. Al igual que el que nos ocupa, el sistema de riego de Iberdrola, mostrado en la figura 2.2, necesita de conexión a Internet para su funcionamiento, pudiendo acceder a ella tanto por conexión WiFi como por vía Ethernet.

Figura 2.2: Controlador de riego de Iberdrola ©Iberdrola. Imagen obtenida de: [14]

Fliwer

Figura 2.3: Controlador de riego de Fliwer ©Fliwer. Imagen obtenida de: [19]

Fliwer combina la inteligencia artificial con la robótica para conseguir un riego lo más eficiente posible. Se puede utilizar tanto en macetas como en terrenos extensos y es válido para riegos de exterior y de interior y se puede combinar con otros dispositivos Fliwer para jardines o plantaciones con más de un tipo de planta diferente. Por otro lado, las opciones de riego están limitadas a césped, flores y huerto. [11]

Con este controlador se puede medir no sólo el caudal de agua, la temperatura y la humedad; si no también el abono, la luz que está recibiendo la planta y la electroconductividad del agua que recibe la misma —capacidad de transportar la electricidad y que puede ser determinante en la cantidad de nutrientes que necesita. El sistema avisará de las necesidades a través de luces de diferentes colores, encargándose automáticamente del riego.

Al igual que el presente sistema de riego, el de Fliwer posee una interfaz web que puede ser accesible desde ordenador, tablet o smartphone. Sin embargo, el módulo controlador no se conecta a la red a través de Ethernet, si no que puede hacerlo de manera inalámbrica, dando al usuario la posibilidad de elegir si a través de WiFi o de tecnología 3G.[19]

Intelliwater de Idis

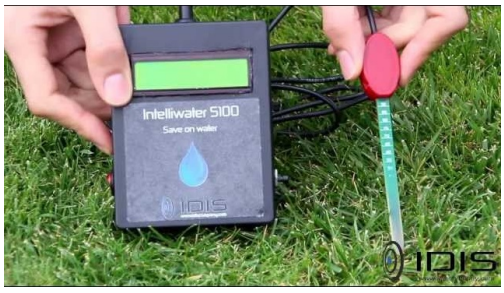


Figura 2.4: Controlador de riego de Idis ©Idis. Imagen obtenida de: [15]

Intelliwater es un sistema que posée varios tipos diferentes de controladores en función del área que se quiera regar. Viene equipado con sensores de humedad, que transmiten los datos al controlador vía radio y que se alimentan con paneles solares, al igual que el sistema de riego propuesto en este trabajo. El sistema Intelliwater, además, consigue un ahorro de agua del 40 %, sin descuidar las necesidades hídricas de cada planta. [15]

Smart Biosystem

Por último, se ha decidido incluir este sistema de riego, orientado a grandes extensiones, por dos razones. Por un lado, al igual que este proyecto, está basado en un software de código libre. La segunda razón es que es el primer sistema en ofrecer un ahorro hídrico de hasta el 60 %, siendo el más avanzado hasta el momento.[34]

Figura 2.5: Controlador de riego de Smart BioSystem ©Smart BioSystem. Imagen obtenida de: [34]

2.3. Marco regulatorio

Actualmente, no hay requisitos regales respecto al uso de sensores o riegos inteligentes en jardines, invernaderos o áreas de cultivo. Todas las restricciones que se encuentran son respecto a la calidad y uso del agua utilizada para regadío y como puede esta afectar a la calidad del alimento en la población. También se tiene en cuenta el uso de aguas residuales tratadas para regadío.

Entre los decretos más importantes que contienen lo anteriormente mencionado se encuentra el **Real Decreto 1/2001, de 20 de julio** —una actualización de la Ley de Aguas—, la **Directiva 2000/60/CE, del Parlamento Europeo y del Consejo** o el **Reglamento (CE) 852/2004**, también con origen en el Parlamento Europeo. [3]

2.4. Entorno socio-económico

Se ha proyectado este trabajo de manera que tenga el menor impacto posible sobre el medio ambiente, siendo ecológicamente sostenible y no generando desechos o residuos de ningún tipo. Para ello, se ha operado en las dos áreas relatadas a continuación:

- **Fuente de alimentación**

  Para alimentar tanto el microprocesador como la electroválvula se ha elegido hacer uso de una energía renovable, en este caso la energía solar. Para ello, como se verá en el capítulo siete, se podrán instalar una placa solar, una batería que permita el máximo ahorro energético y un regulador de tensión. De esta manera, se aprovechan al máximo los recursos del jardín o invernadero donde esté instalado el sistema.

  Para la bomba que suministre el agua no se ha seleccionado una alimentación por energía solar debido a que necesita más potencia. En cambio, de los modelos disponibles se ha elegido uno que funcione con energía eléctrica, en lugar de un motor de combustible diésel o gasolina.

- **Riego por goteo**

  Exceptuando el césped, por razones obvias, para el resto de ambientes se ha elegido un sistema de riego por goteo. Se puede ver en más detalle este sistema en el capítulo siete, pero, en rasgos generales se dirá que se ha elegido este porque proporciona a cada planta la cantidad justa de agua, penetrando hasta la raíz de la misma. Esto no sólo hará que la planta tenga un mejor suministro acorde a sus necesidades hídricas si no que, además, se podrá contar con un considerable ahorro de agua, al utilizar sólo la necesaria.

De este modo, se propone un proyecto que esté lo más adaptado a la sociedad y el medioambiente y, además, suponga un ahorro económico tanto de agua como de tiempo, al evitar al responsable de las plantas tener que estar pendiente del estado del riego constantemente.

Capítulo 3

Herramientas utilizadas

En este capítulo se hablará de las herramientas utilizadas en el desarrollo del proyecto. Se verán, no obstante, sólo las utilizadas para la creación de la página web y del programa, dejando el diseño del sistema completo de riego para ser explicada en el capítulo siete.

3.1. Hardware

3.1.1. EK-TM4C1294XL

Para el proyecto se ha utilizado un mircrocontrolador de Texas Instruments, más concretamente el kit de evaluación EK-TM4C1294XL de la serie Tiva C. Este, a su vez, contiene el microcontrolador TM4C1294NCPDT que viene pre-programado con una aplicación IoT (Internet of Things). Esta aplicación permite, si así se ha programado, subir los datos del programa periódicamente a la nube, gestionada por Exosite[1], y trabajar con ellos desde la plataforma creada por Exosite para Tiva [44].

Esta aplicación resulta muy útil si se quiere hacer tratamiento de datos (por ejemplo de lectura de sensores) y no se posee una página web, si bien ambos recursos, la web y la plataforma Exosite, no son excluyentes el uno del otro. En caso de sí contar con una o varias páginas web, el microprocesador será el encargado de alojarlas, permitiéndo acceder a ellas gracias a una dirección IP proporcionada por el micro, siempre y cuando la extensión de estas páginas no superen la memoria disponible.

Esta memoria se puede clasificar según su tipo, teniendo que el conjunto cuenta con 1 MB de memoria Flash, 256 kB de memoria RAM y 6 kB de memoria EEPROM. Además, se pueden ver otras características del microcontrolador como son: frecuencia de 120 MHz, un ADC SAR de 12 bits y 8 UARTS y 10 I2C que permitirán establecer las comunicaciones en la placa.

Por otro lado, el microcontrolador, como se puede apreciar en la figura 3.1, viene equipado con, entre otras cosas, un puerto Ethernet, un puerto para un conector Micro - USB, 2 botones de usuario, 1 botón de Reset, 4 LEDS y 40 pines donde se pueden conectar

[1] «*Exosite es una plataforma basada en el "Internet de las Cosas"(Internet of Things), que permite a los desarrolladores crear dispositivos conectados con esta plataforma, para así monitorizar los datos obtenidos.*» *[10]*

ampliaciones o módulos de sensores como Sensor Hub BoosterPack; además de numerosos puertos de propósito general que se pueden configurar como puertos de entrada o puertos de salida.

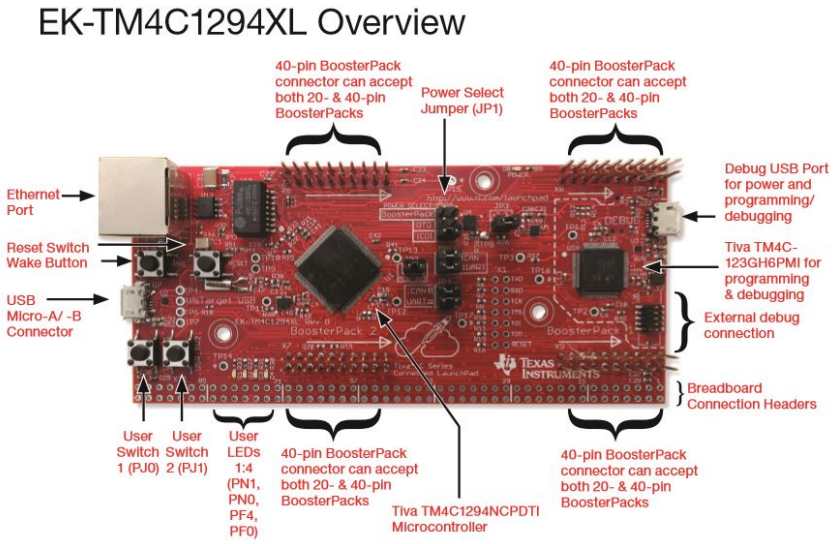


Figura 3.1: EKTM4C1294XL Overview©Texas Instruments. Imagen obtenida de: [38]

Entre las ampliaciones que se pueden añadir al microcontrolador están: un módulo Wi-Fi, que haría innecesaria la conexión mediante puerto Ethernet; una batería, con lo que no se necesitaría estar conectados a alimentación; o un pack de sensores, el Sensor Hub BoosterPack, que es el que se ha utilizado en el proyecto para gestionar los parámetros de los que depende el riego.

3.1.2. Sensor Hub BoosterPack

Sensor Hub BoosterPack es un módulo de expansión para la serie C de Tiva, entre la que se encuentra el microcontrolador que se ha utilizado. Esta ampliación, que se ve en la figura 3.2 y que se podrá colocar en los pines de expansión del microcontrolador, está formado por un total de cinco sensores entre los que se encuentran:

- **Sensor de movimiento MPU9150 :** Compuesto por un acelerómetro de 3 ejes, un giróscopo de 3 ejes y un magnetrómetro de 3 ejes, este es el primer sensor de movimiento de 9 ejes diseñado para baja potencia y a bajo coste. [36]
- **Sensor de presión BMP180:** Se trata de un sensor de presión con un rango de 300 a 1100hPa que, además, puede medir altitud.[2]
- **Sensor de temperatura TMP006:** Este es un sensor infrarrojo que mide tanto la temperatura ambiente como la temperatura de un objeto sin tener que estar en contacto directo. Puede medir de 0ºC a 60ºC con un error máximo de ±1ºC y de -40ºC a 125ºC aumentando ese error a un máximo de ±1,5ºC.[40]

- **Sensor de humedad SHT32:** Se trata de un sensor digital que puede medir humedad con un error del ±2 %, y temperatura, en un rango de -40ºC a 125ºC con un error de hasta ±0,3ºC.[33]

- **Sensor de luminosidad ISL29023:**Se trata de un sensor digital de luminosidad que trata de imitar la respuesta del ojo humano, con un ADC de 16 bits que permite rechazar entre 50 y 60Hz de ruido provocado por la luz artificial.[16]

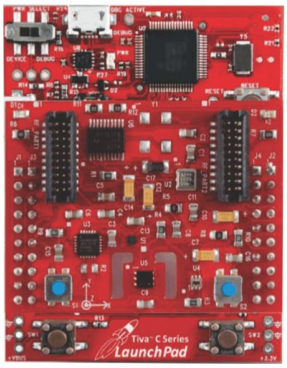


Figura 3.2: Sensor Hub BoosterPack©Texas Instruments. Imagen obtenida de: [37]

Además de los sensores mencionados, Sensor Hub BoosterPack incluye un LED, dos botones y una conexión adicional para el módulo WiFi.

Para la implementación del proyecto, se ha necesitado medir la temperatura y la humedad. Si bien en un primer momento se pensar en usar los sensores de humedad (SHT32) y de temperatura (TMP006), más adelante se eligió medir la temperatura también con el sensor integrado en el SHT32, dado que su margen de error era mucho menor.

3.2. Entornos de desarrollo

3.2.1. Code Composer Studio (CCS)

Code Composer Studio es un entorno de desarrollo integrado creado por Tiva para poder trabajar con sus microcontroladores. Basado en el software Eclipse, incluye un editor de código fuente y un compilador para los lenguajes C y C++. Además, Texas Instruments provee una biblioteca de proyectos, Tivaware, a fin de que el usuario los utilice como ejemplo o como plantilla para desarrollar los suyos propios.[27]

Para el desarrollo de este proyecto se ha utilizado, entre otros, los ejemplos *enet_ io*, *humidity_sht21* y *blinky* de los que se hablará más adelante, en el capítulo 5, cuando se explique el funcionamiento del programa de riego.

3.2.2. NotePad ++

NotePad ++ es un editor de código fuente gratuito que permite trabajar con diversos lenguajes, entre los que se encuentran los utilizados en el trabajo como son: C, HTML, CSS o JavaScript. Se caracteriza por la funcionalidad y la sencillez de uso, distinguiendo en colores las palabras reservadas, comandos y funciones de cada lenguaje, y permitiendo comprimir o expandir el código dentro de funciones para una visualización más sencilla. [39]

3.3. Lenguajes de programación

3.3.1. C

> «*El lenguaje C es un lenguaje para programadores en el sentido de que proporciona una gran flexibilidad de programación y una muy baja comprobación de incorrecciones, de forma que el lenguaje deja bajo la responsabilidad del programador acciones que otros lenguajes realizan por si mismos. Así, por ejemplo, C no comprueba que el índice de referencia de un vector (llamado array en la literatura informática) no sobrepase el tamaño del mismo; que no se escriba en zonas de memoria que no pertenecen al área de datos del programa, etc.*»
>
> (Enrique Vicente Bonet Esteban, n.c.: [8])

Para la realización de este proyecto se ha usado el lenguaje C por varias razones. Una de ellas es su versatilidad respecto a la programación, puesto que al ser un lenguaje tan extendido, existen multitud de bibliotecas desarrolladas por el fabricante para el uso de funciones, lo que hace que el manejo de los periféricos del microcontrolador sea relativamente sencillo; además de las bibliotecas específicas del lenguaje C, que también se han utilizado.

Sin embargo, la razón más importante por la elección de este lenguaje es el interés por poder usar programas ya desarrollados por Tiva como base para el proyecto. Estos programas, de software libre, están incluidos en la anteriormente mencionada Tivaware, y se presentan en su totalidad en lenguaje C, por lo que se ha preferido continuar la programación en este lenguaje.

La resolución del programa de riego está basada el ejemplo *enet_ io* proporcionado por Tiva, el cual se ha tomado como base usando varias de las funciones ya programadas por el desarrollador. Estas funciones, o bien estaban declaradas en los diversos archivos de cabecera (.h), o bien eran funciones declaradas en las bibliotecas <stdint.h>, que se usan para definir los tipos enteros [42]; <stdbool.h>, para trabajar con los valores *bool*, *true* o *false* [41] y <string.h>, que contiene, entre otras, funciones o macros para manipulación de cadenas de caracteres o trabajar con la memoria[43]. Estas bibliotecas, a diferencia de los archivos de cabecera, no son específicos de cada ejemplo y pueden ser utilizados en

cualquier programa con lenguaje C.

Otras bibliotecas son, por ejemplo, *utils* o *driverlib*. Estas, proporcionadas por el fabricante, vienen con los archivos de cabecera de gran cantidad de funciones que permiten trabajar con los periféricos. Por ejemplo, para los pines externos del microcontrolador se tiene, dentro de *driverlib*, el archivo *gpio.h*; y para la UART, dentro de la misma biblioteca, el archivo *uart.h*.

3.3.2. HTML5

HTML es un lenguaje con el que se puede crear una página web básica. Este lenguaje funciona mediante etiquetas, que se forman escribiendo <*nombre de la etiqueta*>, indistintamente en mayúsculas o minúsculas. Salvo excepciones, las etiquetas normalmente van en pares, siendo la de inicio como la que se ha escrito y la del final añadiendo una barra tras el símbolo de menor que. La etiqueta de cierre, por tanto, queda como: </*nombre de la etiqueta*>.[54]

Entre una etiqueta de inicio y una de cierre se puede escribir texto u otras etiquetas, lo que forma estructuras. Algunos ejemplos de etiquetas comunes en una página web son:

- <**html**>**:** Es la etiqueta principal. Con ellas se forma una estructura que contiene el resto de etiquetas de la página web.
- <**head**>**:** Esta etiqueta contiene a las etiquetas <meta> del documento, que son las que, entre otras cosas, proveen información al buscador. También aquí es donde se vinculan las Hojas de Estilo en Cascada (.css) y los ficheros JavaScript (.js), de los cuales se hablará en los siguientes apartados.
- <**title**>**:** Entre estas etiquetas se encuentra el título de la página web. Va ubicada dentro de la etiqueta <head>.
- <**body**>**:** Como su nombre indica, este es el «cuerpo» del documento. Aquí se concentran las etiquetas que darán forma a la parte visible de la página web.
- <**h1**>**:** La etiqueta <h> seguida de un número del 1 al 7 indica que ese fragmento corresponde a una cabecera. Cada número tiene asignado un estilo predeterminado, con una tipología de letra más pequeña en función crece el número. Por ejemplo, se puede asignar una etiqueta <h1> a un título y otra etiqueta <h2> para un subtítulo. A diferencia de las etiquetas anteriormente mencionadas, puede haber más de una etiqueta de cabecera con el mismo número en el documento.
- <**p**>**:** Se pondrá una etiqueta <p> para introducir un nuevo párrafo.
- <**br**>**:** Esta etiqueta sirve para añadir un salto de línea.
- <**a**>**:** Con esta etiqueta se crea un botón. La diferencia entre esta y la etiqueta <button> es que en <a> se puede colocar un atributo, *href* que dirigirá, al pulsar, a otra página web o a un enlace que se elija.

Estas etiquetas pueden, o no, ir seguidas de atributos. Un atributo es un acompañamiento que ayuda a configurar los elementos. Los atributos pueden ser globales o específicos (para todas o una etiqueta concreta, respectivamente). Hay numerosos tipos de atributos. Por ejemplo, se tienen atributos identificativos, como *id* o *name* que permiten poder acceder más tarde a esa etiqueta.

Otros atributos sirven para dar forma y estilo, por ejemplo *font-size*, *background* o *width*, aunque si se quiere dar estilo a toda la página, es más aconsejable hacerlo mediante una Hoja de Estilo en Cascada. Otro ejemplo son los que indican tipo o características: en un formulario se puede poner *required* para hacer un campo obligatorio o *type* para indicar de que tipo se quiere que sea (contraseña, y los caracteres salen ocultos, correo electrónico, y comprueba si es un correo válido... etc).[49]

En caso de ponerlos, sólo se colocarán en la etiqueta de entrada de la siguiente manera: *<nombre de la etiqueta primer atributo="valor del atributo"segundo atributo="valor del atributo">*, separados del nombre de la etiqueta y de los otros atributos por un espacio, y asignándoles el valor con el símbolo de "="seguidos del valor entre comillado.

Para la realización de la página web de este proyecto se ha usado HTML5, la versión más moderna de HTML (2014). La razón para usar es HTML5 es que da soporte a CSS3, con lo que se consigue una apariencia más moderna: usando HTML5 desaparece la cabecera del título tan característica en las páginas web antiguas. Para trabajar con HTML5 tan sólo se tiene que colocar la etiqueta <!DOCTYPE>, que no tiene cierre, en la primera línea del programa, antes que <html>. [22]

3.3.3. CSS

Cascading Style Sheets, u Hojas de Estilo en Cascada en español, es un lenguaje que permite dar forma y estilo a los ficheros HTML a través de las etiquetas o atributos.[20] Hay tres tipos diferentes de CSS:[46]

- **Hoja de Estilo en Cascada externa:** Para crear una Hoja de Estilo externa, es necesario guardar el código en un archivo de extensión .css, y se deberá vincularlo al HTML mediante la etiqueta <link> junto con el atributo *href*, situados dentro de <head>. Este tipo de CSS permite aplicar el mismo estilo a diferentes archivos HTML.
- **Hoja de Estilo en Cascada interna:** En caso de crear una interna, para aplicar ese estilo sólo a una página, se deberá incluir el código dentro de la etiqueta <style>, también dentro de <head>.
- **Estilo en línea:** Este estilo es el mencionado en la sección 3.3.1. y hace referencia a los atributos de estilo. Su uso se aconseja para dar formato a una línea en particular.

Para editar los elementos de la página web, habrá que acceder a ellos nombrándolos en la Hoja de Estilo. Para esto se utilizan los selectores, que se ocupan de encontrar en el documento HTML aquello a lo que se hace referencia. Una vez se haya elegido el selector que más convenga, el código con atributos como *position*, *background* o *font-family*, dependiendo de las necesidades, irá entre llaves a continuación. Se puede distinguir tres clases de selectores principales:[47]

- **Selector de elemento:** Para seleccionar un elemento, se usará su nombre de etiqueta. Por ejemplo, si se pone *p{(código)}*, se podrá dar estilo al el elemento <p> del archivo HTML. Sin embargo, en caso de haber más de una etiqueta con ese nombre en el archivo, el estilo se aplicará a todas.
- **Selector de id:** Id es un atributo global(esto sólo se aplica para HTML5), que debe ser único en todo el documento HTML y que permitirá identificarla. Para seleccionar una etiqueta por su id se utilizará la almoahdilla #, de tal modo que queda: *#elemento1{(código)}*.[53]
- **Selector de clase:** Para seleccionar por clase, previamente se deberá haber incluido el atributo global *class*. Este se diferencia del atributo *id* en que puede haber más de una etiqueta con la misma *class*. Para seleccionar una o varias etiquetas, se usará el punto. Por ejemplo: *.clase1{(código)}*.[50]

También se podrán agrupar selectores del mismo tipo, separándolos mediante comas y poniendo el código entre llaves tras el último: *.clase1, .clase2, clase3 {(código)}*. Además, se podrá acceder a la clase de un determinado elemento escribiendo esta justo a continuación: *elemento.clase1 {(código)}*.

Al igual que pasaba con HTML5, «CSS3 es la última evolución del lenguaje de las Hojas de Estilo en Cascada (Cascading Style Sheets), y pretende ampliar la versión CSS2.1. Trae consigo muchas novedades altamente esperadas, como las esquinas redondeadas, sombras, gradientes, transiciones o animaciones, y nuevos layouts como multi-columnas, cajas flexibles o maquetas de diseño en cuadrícula (grid layouts).» [21]

3.3.4. JavaScript

JavaScript es un lenguaje orientado a objetos conocido por aportar a las páginas web animación y posibilidad de interacción con el usuario, aunque también puede ser usado en entornos sin navegador. [23]

Si se quiere incluir dentro de la página web, se podrá hacer de dos maneras. La primera consiste en crear un fichero .js —donde se incluirán las funciones en lenguaje JavaScript que se necesite utilizar— y vincularlo al código HTML mediante las etiquetas <script> y </ script> y el atributo *src* en el apartado <head>. Este método es muy útil si se tienen varias páginas que deben compartir datos o usar la misma función.

La segunda opción para usar JavaScript en el trabajo es incluirlo dentro del código HTML. Se podrá hacer esto escribiendo el código entre las etiquetas <script> y </script> dentro del <body>.

Para poder llamar a las funciones en el código, existen los llamados Eventos HTML DOM [51], que se colocan dentro de las etiquetas de la misma manera que los atributos. Así, se consiguen funciones que se activen cuando se pulsa con el ratón *(onclick)*, cuando se pulsa una tecla en el teclado y, posteriormente, se suelta *(onkeyup)*, cuando se pasa con el ratón por encima *(onfocus)* o cuando se clica o se desclica *(onchange)*, entre otras muchas opciones.

En este proyecto se ha usado JavaScript como complemento en la aplicación web y, en entornos sin navegador, como función de comunicación entre esta y el microprocesador, lo que se explicará más detalladamente en los capítulos 4 y 5 respectivamente.

Capítulo 4

Diseño de la web

En este capítulo se detalla la creación de la web que servirá tanto para recoger las preferencias del usuario con respecto del ambiente seleccionado, como para informar al mismo de las condiciones de riego.

La razón de crear una página web y no interactuar directamente con el programa a traves del puerto UART responde a dos siguientes objetivos:

- Proporcionar al usuario un entorno fácil y conocido, con el que se encuentre, en mayor medida, familirizado. De este modo es seguro que el programa llegue a todos los públicos, y no sólo a gente acostumbrada a tratar con terminales de puerto serie.
- Interactuar con el sistema a través de cualquier dispositivo que tenga un navegador web, es decir, además de poder usar un ordenador, poder acceder al estado de los riegos a través de un smartphone o una tablet, algo que no sería posible de haber implementado la comunicación mediante UART.

Respecto al estilo de la web, dado por las Hojas de Estilo en Cascada (CSS), para el desarrollo de este proyecto se ha creado seis archivos .html diferentes, cada uno con su propia Hoja de Estilo en Cascada Externa. Esto se ha hecho para poder manejar ambos códigos de manera más sencilla, en lugar de tener que trabajar con un fichero de una extensión demasiado larga. Por otra parte, para poder adaptar la web a diferentes dispositivos, navegadores y/o diferentes estados de zoom, se debió de cambiar todo el diseño de Hojas de Estilo en un punto avanzado del proyecto, puesto que en en la versión inicial todas las medidas y distancias se definían como un porcentaje relativo a otra medida, en lugar de ser un valor fijo, y eso provocaba severos cambios en la apariencia de las páginas.

Además de las respectivas Hojas de Estilo en Cascada, también se ha incluido JavaScript en un fichero vinculado a cuatro de las páginas, cuya función se explicará más detalladamente en cada una de ellas.

Sin embargo, como la web debe ir alojada en el microprocesador, hay que tener en cuenta una serie de impedimentos. Por un lado, no se puede usar PHP ni jQuery —un lenguaje de programación y una librería de JavaScript, respectivamente, que añadirían

versatilidad a la web— dado que el microprocesador no puede soportarlos y, al cargar el programa, no funcionan como deberían.

Otra restricción a tener en cuenta es el tamaño. Debido a la memoria del microprocesador, el conjunto de páginas web,imágenes alojadas, hojas de estilo y ficheros JavaScript no debían superar los 1024 kB. Debido a esto, se tuvo que retirar la imagen de fondo inicial y ser sustituida por un color uniforme, además de ser otra razón para prescindir del uso de jQuery. Actualmente, el conjunto total de ficheros incluidos suma 114 kB.

Por otro lado, a pesar de haber creado una Hoja de Estilo para cada archivo .html, todas las páginas presentan elementos comunes. En primer lugar, se tiene el pie de página, situado al final de la misma, que muestra el nombre de autor y una licencia común. Además de esto, otro elemento presente en todas las páginas es el escudo de la universidad, situado, excepto en la página principal —donde será la cabecera— en la esquina superior derecha de la interfaz. Si en cualquier momento se pincha sobre el escudo, se abrirá la página de inicio de la UC3M en una nueva pestaña del navegador.

Cabe destacar que este conjunto de páginas web, con sus correspondientes Hojas de Estilo, se ha hecho expresamente para la resolución de este problema en concreto. En caso de que cualquier otra compañía ajena a la Universidad Carlos III de Madrid estuviese interesada en utilizar esta interfaz, tanto el escudo de la universidad, como los colores (en caso de querer poner los colores propios de una interfaz determinada), como también los datos de contacto (ver sección 4.2.) son fácilmente modificables de manera que se pueda adaptar a la compañía encargada de la gestión y mantenimiento del problema.

A continuación, se explicará más en detalle cada una de las páginas que forman la web[1], cuyo código se puede ver detalladamente en el Anexo 3, al final de esta memoria.

4.1. Inicio

La mostrada en la figura 4.1 es la página que aparece cuando se introduce la dirección IP del micro en el navegador. Para ello, se ha nombrado como *index.html* de tal manera que será reconocida como la página web inicial. En este caso, la página de inicio se trata de una página web simple con tres botones que conducen a otras tres páginas diferentes.

[1]Las imágenes mostradas se corresponden con las páginas en el navegador Mozilla Firefox. En caso de utilizar otro navegador, el estilo de letra, cuadros de alerta o cuadros de selección podría verse modificado.

Figura 4.1: Página de inicio

Para el funcionamiento de esta página no se ha utilizado JavaScript, puesto que ha bastado la etiqueta <*input*> con el atributo *href* para acceder a la página que se haya seleccionado.

Por otro lado, el estilo mostrado es el que se sigue en todas las demás páginas, exceptuando la página principal. Se trata de un fondo verde en tonos pastel, un cuadro negro semitransparente sobre el que se mostrará el menú y el escudo de la universidad en la esquina superior derecha. Al igual que el resto de botones de la aplicación, estos se oscurecen al pasar el ratón sobre ellos o hacer clic.

4.2. Contacto

Si en la página de inicio se pulsa el botón *Contacto*, se accederá a la página que se ve en la figura 4.2. Se trata de una página cuya función es meramente informativa, razón por la cual, como la anterior, no hay necesidad de usar JavaScript, puesto que la única interacción que tiene el usuario con dicha página se produce con el botón *volver*. Para la implementación del mismo también se ha usado la etiqueta html <*input*> con el atributo *href*.

La función de *Contacto* es mostrar los datos del trabajo.En este caso se ha decidido mostrar el nombre, la organización o el correo de contacto, pero, como se mencionó anteriormente, los datos aquí mostrados son fácilmente configurables.

Figura 4.2: Página de contacto

4.3. Unirse

Al pulsar el botón *Unirse*, se reconducirá al usuario a la página mostrada en la figura 4.3.Aquí, se deberá introducir el nombre de usuario y la contraseña que elegida al registrarse. En caso de no haberlo hecho, o no recordar las credenciales, se deberá pulsar sobre el botón para unirse, que conducirá a la página de *Registro*.

Figura 4.3: Página de acceso

Si el usuario se ha registrado, pero introduce mal sus datos de usuario, una ventana emergente como la de la figura 4.4 alertará de que estos se deben revisar para poder acceder. Cuando estos datos sean correctos, se redirigirá al usuario a la página principal, donde se muestra el estado del riego.

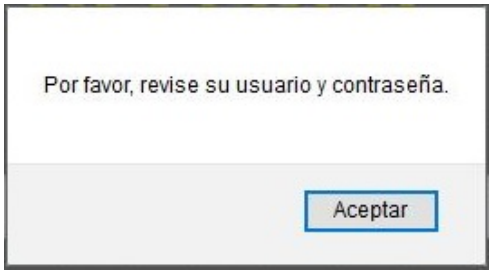


Figura 4.4: Error al acceder

Para la realización del formulario de entrada se ha usado[2] las etiquetas <*input*> para determinar que estos campos corresponden con cuadros de entrada de datos. Se ha completado con los atributos *required*, para hacer ambos campos obligatorios, y *type="password"* en la etiqueta <*input*> correspondiente a la contraseña para que de esta manera los caracteres permanezcan ocultos al escribir.

Para la comprobación de los credenciales, se ha usado una función JavaScript que se activa al pulsar el botón «Acceder». Esta función obtiene los valores guardados tras el registro —para lo cual usa la función *localStorage.getItem* igualada a una variable— y los compara con los introducidos — habiéndolos guardado en otra variable gracias a *document.getElementById* [35]. La comparación se realiza mediante un *if*, para lo cual deberá trabajar con los valores de ambas variables, lo que hará comparando en el *if* de la siguiente manera: *variable1.value==variable2.value*.

Si los datos introducidos resultan correctos, usa la función *window.location.href* para reconducir a la página principal. En caso de no serlo, la función que usa es *window.alert*, que creará una ventana emergente con un mensaje. Para el botón que lleva a la página de registro se ha utilizado el atributo *href*.

4.4. Registro

En caso de que, en la página principal, se haya pulsado el botón *Registro*, se accederá a un formulario típico de registro como el mostrado en la figura 4.5. Si lo que se quiere es acceder con un usuario ya registrado, al igual que en la página de acceso, se tendrá la posibilidad de cambiar con el botón situado en la esquina inferior del cuadro.

[2]Ver Anexo 3 para mayor entendimiento del código.

También se deberá acceder a esta página de registro en caso de querer cambiar el tipo de ambiente o de terreno del programa de riego. Sin embargo, podrán volver a introducirse los mismos datos de usuario si así se desea.

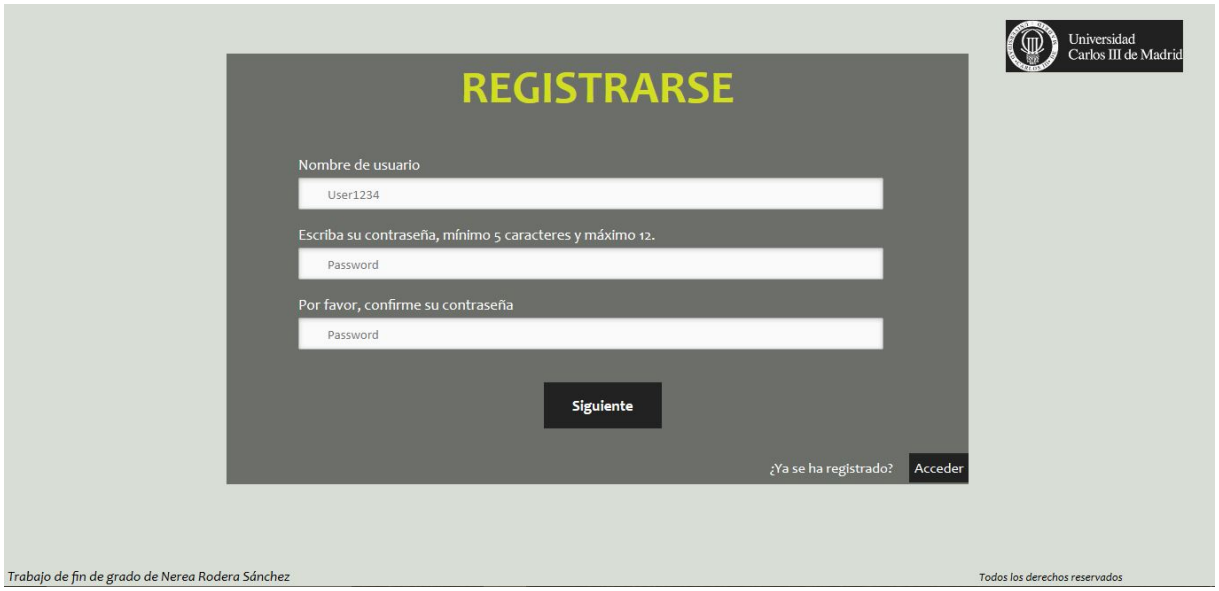


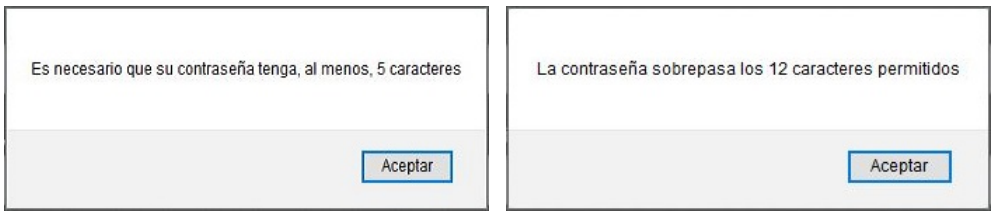
Figura 4.5: Página de registro de usuario

La finalidad de esta página es elegir un nombre de usuario que contenga al menos un carácter y una contraseña para poder acceder cada vez que el usuario se registre. Para ello, se deberá elegir una contraseña que tenga entre cinco y doce caracteres y repetirla en el cuadro siguiente para asegurar haberla escrito correctamente. En caso de no cumplir con estas contraseñas aparecerán con los errores mostrados en la figura 4.6 —contraseña incorrecta— o en la figura 4.7 —usuario incorrecto.

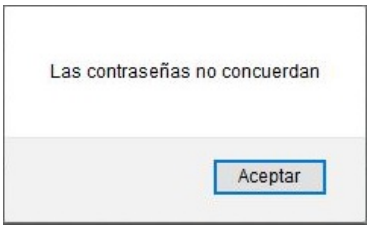
Al igual que en la página de acceso, se han usado las etiquetas <*input*> junto con los atributos *required* y *type="password"* en la entrada de datos. También al igual que en la anterior, se ha usado *href* en el botón que permite acceder directamente sin pasar por el registro.

Por otro lado, si en la página de inicio (ver figura 4.5) se pulsa *Siguiente*, se llamará a una función de JavaScript que comprobará en primer lugar la validez del usuario introducido. Para ello se usará la función *document.getElementById.value*, que almacenará el usuario en una variable.

Posteriormente, se comparará la longitud en caracteres del usuario con la longitud mínima, que en este caso se le ha puesto sólo un carácter. Para realizar esta comparación se deberá medir la longitud de la variable donde se ha guardado el usuario mediante *variable.lenght* y compararla con el número de caracteres gracias a un if.



(a) Error: no supera los 5 caracteres (b) Error: supera los 12 caracteres



(c) Error: contraseñas no coinciden

Figura 4.6: Errores al registrar la contraseña

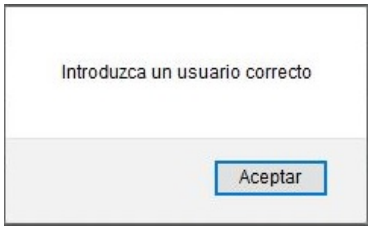


Figura 4.7: Error al registrar el usuario

En caso de que el usuario sea correcto, se procederá a guardar la contraseña y la comprobación en dos variables, sólo usando *document.getElementById* para posteriormente comparar sus valores con un *if(variable1.value==variable2.value)*. En caso de ser ambas correctas, se procederá a acotar la contraseña entre 5 y 12 caracteres de la misma manera que se ha hecho con el usuario. Si tras estas comprobaciones todo está correcto, la función *window.location.href* ns redirigirá a la segunda parte del registro, el formulario. De no ser así, aparecerá una ventana con el error correspondiente.

4.5. Formulario

Si ya se ha elegido un usuario y contraseña en la página de registro, se accederá al formulario que permitirá seleccionar el tipo de planta y de terreno que regará el programa.

Para escoger el tipo de planta, se contará con nueve cuadros de selección, o *checkboxes*, divididos en seis ambientes de plantas de exterior y tres de plantas de interior (la separación de los ambientes corresponde a la localización habitual de las plantas en un contexto doméstico y, por tanto, no afecta a la ubicación de la plantación y del sistema de riego —invernadero o plantación exterior—).

Además, se permitirá elegir entre los dos tipos más comunes de terreno, para lo que el programa provee una lista desplegable de selección donde se podrá elegir entre *terreno árido* y *terreno húmedo*.

A continuación, en las figuras 4.8 y 4.9 se muestra el aspecto de la página de formulario[3].

Figura 4.8: Formulario de selección de parámetros de riego (1)

Figura 4.9: Formulario de selección de parámetros de riego (2)

[3]La extensión de la página es demasiado grande para mostrarse en una sola imagen.

Para realizar los cuadros de selección se ha utilizado la etiqueta <*input*> acompañada del atributo *type="checkbox"*. Los cuadrados de colores han sido realizados en la Hoja de Estilo adjunta. Por otro lado, para conseguir la lista desplegable se ha hecho uso de la etiqueta <*select*> y colocando cada opción entre las etiquetas <*option*> , dentro de la estructura de selección.

Además, se ha incluido un botón que permite finalizar el registro y acceder a la página principal. Este botón llama a otra función de JavaScript que se ocupa, por un lado, de comprobar si la opción seleccionada de la lista es la opción en blanco y, por otro, de contar el número de ambientes seleccionados.

En el primer caso, se comprobará si se ha seleccionado alguno de los dos tipos de terrerno mediante el método *document.querySelector*, que devuelve el estado de un elemento por su clase [52]. Para saberlo, se obtendrá su valor *document.querySelector(".class").value* y se comparará con el valor nulo de la lista desplegable " ". Si el resultado de esta comparación da positivo, se mostrará un mensaje como el de la figura 4.10.

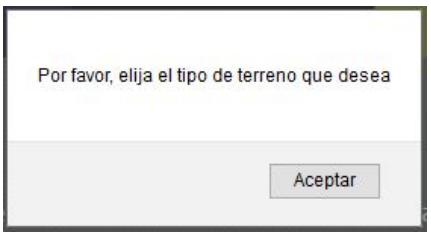
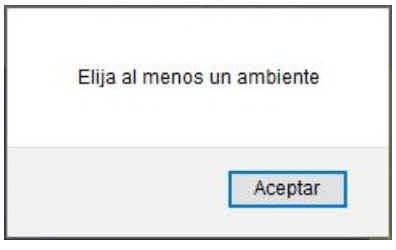
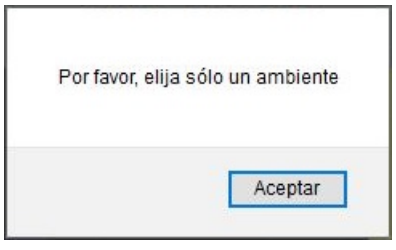


Figura 4.10: Error: no se ha seleccionado tipo de terreno

En el segundo caso, para comprobar si un ambiente está seleccionado usa *if (document.getElementById("checkbox1").checked)* en cada cuadro, incrementando en uno la variable usada como contador en caso de que la condición sea verdadera. Posteriormente se comprobará el valor del contador; si no llega a un ambiente seleccionado , o supera este, mostrará, respectivamente, los mensajes mostrados en las figuras 4.11(a) y 4.11(b).



(a) Error: no hay ambiente seleccionado



(b) Error: supera el número de ambientes máximo

Figura 4.11: Errores en la selección de ambientes

Además de la comprobación de los parámetros, cada vez que se selecciona un cuadro o se elige una opción de la lista de selección se llama a su correspondiente función en

lenguaje JavaScript que permitirá comunicar la web con el programa de riego. Esto se verá con mayor detalle en el capítulo siguiente.

4.6. Página principal

La página principal, a la que se puede acceder tras rellenar el formulario o desde la página de acceso, es la más importante del conjunto de páginas que forman la web de este trabajo, dado que muestra en tiempo real las condiciones climáticas y el estado del jardín, invernadero o zona de riego. Por esta razón, se le ha querido dar un aspecto diferente al del resto de páginas.

Como se puede ver en la figura[4] 4.12, el conjunto de la página se centra en el cuadro central, que da información de:

- **Parámetros seleccionados en el formulario de registro:** Tales como el ambiente elegido, si se trata de un formulario de exterior o interior y el tipo de terreno que se ha seleccionado.
- **Parámetros proporcionados por el programa de riego:** Como los datos meteorológicos obtenidos de los sensores (temperatura, humedad y temperatura dependiente de la humedad) o los datos conseguidos tras el procesamiento de los anteriores (estado del riego y tiempo hasta el próximo riego).

Además de esta información, completa la página principal un botón de *Cerrar sesión* que llevará de vuelta a la página de inicio. En una primer momento, se pensó añadir un botón que permitiese volver al formulario y reelegir el ambiente. Sin embargo, debido a la cantidad de funciones que tenía que llevar a cabo la página, esta solución no aportaba resultados óptimos, por lo que se prefirió acceder al cambio de ambiente a partir de un nuevo registro.

También se puede ver, en la esquina superior izquierda de la página, el nombre de usuario con el que el usuario se ha registrado, la fecha en formato dd/mm/yy, y la hora en formato hh:mm:ss. Esto se ha conseguido gracias a funciones en lenguaje JavaScript. Para el primer caso, simplemente se ha imprimido en pantalla la variable donde, en *Registro*, se almacena el nombre de usuario.

Para imprimir la fecha y la hora, el programa está basado en un ejemplo de la página W3 [48] para mostrar la hora y, modificada con las funciones correspondientes —*getDate()*, *getMonth()* y *getFullYear()*—, la fecha. Esto se podría haber logrado con la función *Date()*, sin embargo, suponía una pérdida de versatilidad dado que se aplicaba la representación UTC [55].

[4]Esta imagen corresponde a una situación específica. Los datos mostrados pueden variar en función del ambiente escogido y las medidas de los sensores. La imagen ha tenido que ser redimensionada para poder apreciar el contenido de la página.

Figura 4.12: Página principal del programa

Para poder llamar, al recargar la página, a múltiples funciones sin que superpusieran, se ha utilizado la función *addLoadEvent*, una función obtenida en [13], que llama de uno en uno todas las funciones que se haya seleccionado mediante el método *window.onload* de JavaScript.

Por otro lado, también con JavaScript, se muestra un mensaje al cargar la página como el de la figura 4.13. Esto se debe a que, al cargar la página, se llaman muchas funciones JavaScript, tanto para mostrar los datos en pantalla, como para enviar órdenes al programa. Cada función JavaScript envía una petición web al servidor, en este caso el microprocesador, provocando que esto ralentice la carga de datos en la página. Por eso,

algunos datos pueden tardar unos segundos en mostrarse, dependiendo de lo ocupado que se encuentre el microprocesador y de la conexión Ethernet.

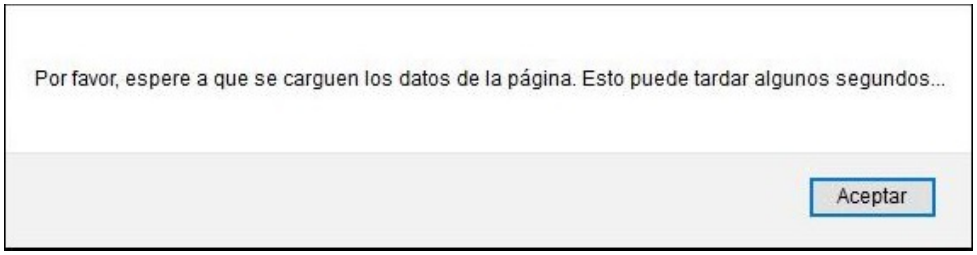


Figura 4.13: Mensaje de espera en la página principal

Por último, en referencia al estilo de la página, en esta se ha preferido mostrar una página diferente a las demás. Para ello, el principal cambio se puede notar en el fondo, ahora de color blanco, en lugar de verde. Otra diferencia notable es el escudo de la universidad, normalmente situado en la esquina superior derecha de la página, ahora se encuentra en el centro haciendo las veces de cabecera para la página. Así mismo, el cuadro contenedor también ha cambiado, pasando de ser negro semitransparente a un tono azul oscuro.

Capítulo 5

Programa enet_io

A continuación, se comenzará con la explicación del programa introducido en el microprocesador, el cual se puede ver con detalle en la figura 5.1.

Se ha decidido dividir la explicación del programa entre este capítulo y el siguiente, el capítulo 6. Esto se ha hecho tanto por motivos de extensión, como de procedencia — todo el código aquí tratado se ha adaptado de los ejemplos de código libre presentes en la librería *Tivaware*. También por esta razón se ha decidido no incluir el código de los apartados a continuación desarrollados, puesto que se pueden encontrar en la librería, a diferencia del código del programa de riego, que se verá en el Anexo 4.

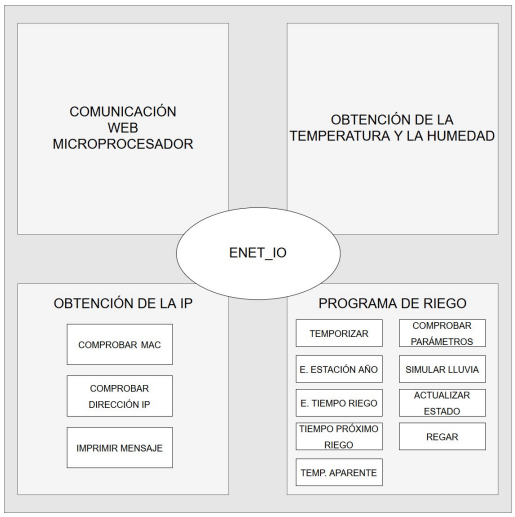


Figura 5.1: Esquema del programa

En esta figura, se puede ver que el programa, *enet_io* se ha dividido en cuatro subsecciones. Exceptuando la última, el programa de riego (que se explicará con más detenimiento en el capítulo 6), el resto provienen de ejemplos que proporcionaba Tiva en su librería y que se han adaptado para el correcto funcionamiento del programa. A continuación se verá cada subsección del programa con más detalle.

5.1. Comunicación web-microprocesador

Como ya se explicó en el capítulo 3, la web que se ha creado se aloja dentro del microprocesador, y se comunica a través del lenguaje JavaScript.

Se comenzará explicando cómo hacer que el microprocesador actúe de servidor. Para ello, el programa se ha basado en el ejemplo *enet_io*, incluido en la librería *TivaWare* proporcionada por Tiva.

En primer lugar, tras incluir la carpeta con el programa en el directorio de trabajo, se accederá a la carpeta *fs*, donde se almacena todo el contenido asociado a la web (páginas, hojas de estilo, ficheros JavaScript e imágenes). Allí, sustituimos los archivos de ejemplo por los del proyecto, siempre cuidando que no sobrepasen la memoria máxima para evitar problemas de compilación.

Una vez hecho esto, el siguiente paso es modificar el archivo *io_fsdata.h*, que es el archivo de imagen del sistema y se halla dentro de la carpeta del programa. Si no se hace esto, aunque se haya modificado la carpeta *fs*, el programa seguirá mostrando las páginas antiguas. Para modificar este archivo y así poder actualizar el programa con la web, se deberá hacer uso del archivo *makefsfile.exe* ubicado en el directorio donde se haya descargado la librería *TivaWare*. Esta se encuentra dentro de la carpeta *bin*, que está a su vez dentro de la carpeta *tools*. [1]. Este es el archivo que creará automáticamente un nuevo *io_fsdata.h*.

Una vez localizada la aplicación *makefsfile.exe*, la siguiente tarea es abrir la consola de comandos (cmd) de Windows. Allí, se introducirá el comando *cd* seguido de la dirección de la carpeta del programa, que deberá estar incluida en el «workspace» o directorio de trabajo, y se pulsará intro. A continuación, se deberá escribir el directorio donde está el archivo *makefsfile.exe* y, tras ello, la línea *-i fs -o io_fsdata.h -r -h*. Esta viene dada en el archivo de instrucciones (*readme*) y en los comentarios de introducción del programa.

Tras la creación del nuevo archivo de imagen se deberá volver a «construir» y compilar el programa desde Code Composer Studio, para introducir los nuevos cambios.

Respecto a la comunicación, esta se hace bidireccionalmente, es decir, se pueden enviar tanto datos de la web al microprocesador (ambientes seleccionados, tipo de terreno elegido), como del microprocesador a la web (información de los sensores, estado del riego). Para ello, se deberá hacer uso de la tecnología CGI, que utiliza el lenguaje JavaScript. Así, se tienen dos tipos de funciones, íntimamente relacionadas, que se encuentran en la web (en el fichero JavaScript o dentro de las etiquetas *<script>)* en el propio archivo html) y en el programa cargado al microprocesador, más concretamente en el archivo *io_fs.c*.

La primera se puede ver en la figura 5.2 donde hay una función ubicada en el fichero JavaScript. Esta que se muestra de ejemplo, es concretamente, la función que se activa

[1] Esta aplicación sólo funcionará si se está trabajando con un Windows de 32 bits. En caso de hacerlo con uno de 64, se deberá descargar otra versión del archivo que Tiva pone a nuestra disposición.

si se ha pulsado el primer cuadro de selección en el formulario y que, además, tendrá su «función espejo» en el archivo *io_fs.c*, la cual se puede ver en la figura 5.5.

```
function checked1()
{
    var req = false;

    function amb1Complete()
    {
        if(req.readyState == 4)
        {
            if(req.status == 200)
            {
                document.getElementById("amb1state").innerHTML = "<div>" +
                                                    req.responseText + "</div>";
            }
        }
    }

    if(window.XMLHttpRequest)
    {
        req = new XMLHttpRequest();
    }
    else if(window.ActiveXObject)
    {
        req = new ActiveXObject("Microsoft.XMLHTTP");
    }
    if(req)
    {
        req.open("GET", "/cgi-bin/checkbox1?id" + Math.random(), true);
        req.onreadystatechange = amb1Complete;
        req.send(null);
    }
}
```

Figura 5.2: Ejemplo de función JavaScript de transferencia de datos

En esta función se prestará especial atención a dos líneas. La primera se puede ver en la imagen 5.3 y es el código que, una vez ha obtenido respuesta del servidor, imprime esta en el lugar que se le haya indicado, entre dos etiquetas * *.

```
document.getElementById("amb1state").innerHTML = "<div>" +
                                    req.responseText + "</div>";
```

Figura 5.3: Línea de código de la función JavaScript de transferencia de datos (1)

La segunda es la representada en la figura 5.4 y corresponde a la petición de la web al servidor. Se debe prestar especial atención a */cgi-bin/checkbox1?id"*, puesto que esta será la condición de entrada del else que, situado en el archivo *io_fs.c* (figura 5.5), permitirá actuar sobre los datos recibidos.

```
req.open("GET", "/cgi-bin/checkbox1?id" + Math.random(), true);
```

Figura 5.4: Línea de código de la función JavaScript de transferencia de datos (2)

En la siguiente imagen se puede ver la función *ustrncmp*, que compara dos cadenas, y que dirá si se ha llamado a esta función. En caso de que la función el valor «0» significará que la comparación ha sido positiva y se realizarán las asignaciones y operaciones que se han programado dentro del else if.

```
else if(ustrncmp(pcName, "/cgi-bin/checkbox1", 20) == 0)
{
    static char pcBuf[12];

    tipo1=1;
    tipo2=0;
    tipo3=0;
    tipo4=0;
    tipo5=0;
    tipo6=0;
    tipo7=0;
    tipo8=0;
    tipo9=0;

    psFile->data = pcBuf;
    psFile->len = strlen(pcBuf);
    psFile->index = psFile->len;
    psFile->pextension = NULL;

    //
    // Return the psFile system pointer.
    //
    return(psFile);

}
```

Figura 5.5: Ejemplo de función en el archivo fs_io.c (1)

Por otro lado, si se quiere devolver información a la página web y que esta imprima algún carácter, se puede hacer con la función *usnprintf*, como se muestra en la figura 5.6, que imprimirá el texto entre comillas en el lugar que se le haya indicado mediante el atributo *id* (ver imagen 5.3).

Al igual que se ha decidido imprimir texto, se pueden enviar variables con números o caracteres, o una combinación de ambos, como se muestra en la figura 5.7, correspondiente al envío de horas restantes hasta el próximo riego. Esta, y otras variables que se necesitan mostrar en pantalla (como por ejemplo la temperatura o la humedad), varían cada cierto tiempo. Por eso, es necesario llamar a las variables no solamente al cargar la página sino cada cierto tiempo. Para ello se utiliza la función *setTimeout*, que, incluida dentro de la función de petición al servidor en la hoja JavaScript, permitirá reevaluar o volver a llamar a la función tras un tiempo que se le asignado (en milisegundos) [56].

```
usnprintf(pcBuf, 50, "%d HORAS",horas_hasta_riego);
```

Figura 5.7: Línea de código de la función JavaScript de transferencia de datos (3)

```
else if(ustrncmp(pcName, "/cgi-bin/display_terr", 22) == 0)
    {
        static char pcBuf[12];

        if(terreno1==1)
                {
                    usnprintf(pcBuf, 12, "ARIDO");
                }
        else
                {
                    usnprintf(pcBuf, 12, "HUMEDO");
                }


        psFile->data = pcBuf;
        psFile->len = strlen(pcBuf);
        psFile->index = psFile->len;
        psFile->pextension = NULL;

        //
        // Return the psFile system pointer.
        //
        return(psFile);

    }
```

Figura 5.6: Ejemplo de función en el archivo fs_io.c (2)

Sin embargo, aunque se tiene la posibilidad de imprimir tanto variables como constantes o cadenas de caracteres, la función *usnprintf* no permite imprimir variables *float*. Por lo tanto, en caso de querer imprimir números con coma, se deberá separar la parte entera de la decimal e imprimir ambos con un punto o coma separándolos. Por otro lado, esta función tampoco admite ciertos caracteres ASCII como el ASCII Extendido, por lo que no se puede poner tildes sobre las letras, puesto que estas no se imprimirían. Por esta razón, se ha decidido imprimir los mensajes pertinentes en mayúsculas.

Cabe destacar que el método de las funciones en JavaScript no es el único que proporciona Tiva en sus ejemplos. En este mismo, el ejemplo *enet_io* de la librería *TivaWare*, explican que se pueden transmitir los datos a través del lenguaje CGI o bien a través de la creación de archivos SSI (Server Side Includes) que son unas rutinas programadas en HTML, pero con la extensión del archivo en .ssi [57]. No se ha explorado este método porque el anterior pareció más intuitivo y más adaptable al modelo de programa de funciones que se quería seguir, además de poder continuar así la programación en lenguaje JavaScript.

Por último, se debe añadir que todas las funciones utilizadas para el propósito de comunicación entre el servidor y la web han sido adaptadas de los ejemplos del programa *enet_io*, y, por tanto, se ha decidido mantenerlas tal cual, para que, en caso de querer modificar el programa o una persona externa trabajar sobre él, sea más sencillo. Por esta razón, hay algunas funciones que no transmiten datos «de vuelta» a la web pero que, aun así, se han mantenido líneas de código como las mostradas en la figura 5.3.

5.2. Programa de medición de la temperatura y la humedad

Como se comentó en el capítulo 3, en un principio se pensó en usar dos sensores diferentes, uno para la temperatura y otro para la humedad, eligiendo finalmente usar sólo el sensor de humedad. Con esto, se aseguraba una medida más precisa (al ser el error más pequeño).

Para obtener las mediciones de los sensores, en un principio se pensó en tomar de base el ejemplo *senshub_iot*. Este es un programa de Tiva que reúne todos los sensores, junto con sus funciones de medición de datos, escritura y calibrado (en caso de necesitarlo) y todas sus interrupciones. Además de esto, *senshub_iot* ofrece conexión a la nube y subida periódica de los datos a la plataforma Exosite de Tiva.

Sin embargo, la unificación de este programa con el ejemplo *enet_io* no dio resultado. Además, puesto que no se iban a utilizar las funcionalidades del programa al cien por cien (dado que con la página web no se necesitaba de la plataforma Exosite y que no se iban a utilizar todos los sensores), se prefirió utilizar el programa de cada sensor.

Cada sensor incluido en *Sensor Hub BoosterPack* tiene su propio ejemplo independiente en la librería *TivaWare*. Así pues, en un principio se pensó en utilizar los programas *temperature_tmp006* y *humidity_sht21* —temperatura y humedad, respectivamente—, quedando finalmente sólo el último.

Para poder hacer uso de él, en primer lugar se debe vincular el archivo .c al programa. Para eso, simplemente, el programa abierto en Code Composer Studio, se debe seleccionar el archivo .c y arrastrarlo. Code Composer preguntará si se quiere copiar o vincular y, si bien ambas opciones son válidas, en este caso se ha elegido crear una copia, para así poder trabajar sobre ella y modificarla sin afectar al original.

Entonces, el siguiente paso será configurar las interrupciones de las que hará uso el sensor de humedad. Para ello, se abre el archivo *startup_css.c* y, junto a las declaraciones de interrupciones externas usadas por la aplicación, se copiará la correspondiente al sensor: *extern void SHT21I2CIntHandler(void);*. Esta se puede encontrar en el archivo *startup_css.c* del programa del sensor de humedad.

También en *startup_css.c*, pero en este caso en la tabla de vectores, se deberá modificar una línea. La interrupción *I2C7 Master and Slave* está marcada como interrupción por defecto (*IntDefaultHandler*); habrá que cambiarla por *SHT21I2CIntHandler*.

Una vez hecho esto, hay que fijarse en que se tienen dos funciones *main*, la de *enet_io* y la de *humidity_sht21*. Como en este caso se ha elegido que *enet_io* sea el programa principal, se ha cambiado el *main* del programa del sensor de humedad, de tal modo que ha pasado de *int main (void)* a *void hum_task (void)*. De esta manera, es una función que no retorna valor y a la cual se puede llamar.

Para llamarla, se ha creado mediante JavaScript una función que se activa al cargar la página y, a partir de ese momento, cada segundo. Esta función manda una petición al servidor que, cuando la recibe, llama a la función de humedad. Allí, haciendo uso de las librerías y funciones de Tiva, se leen la temperatura y la humedad, se almacenan en variables —cuatro variables: entero y decimal para la temperatura, y entero y decimal para la humedad— y se procede a mostrarlas por pantalla y a trabajar con ellas en el programa de riego de la manera que se explica en el punto siguiente.

5.3. Obtención de la IP

Otro de los factores importantes en el programa *enet_io* es la asignación de una dirección IP que permita conectar el servidor con la página web. Existen dos maneras de hacerlo: la primera es asignar una dirección IP estática manualmente. Para ello, se deberá conocer la dirección del dispositivo donde se conectará el microprocesador.

La otra manera, que es la usada por Tiva en el ejemplo y que se ha preferido no modificar, es la obtención de una IP dinámica mediante DHCP. Este es un protocolo de comunicación entre el servidor y el cliente, y que permitirá obtener la dirección IP a la que se esté conectado. Se ha elegido esta opción para facilitar al usuario la tarea de conexión, evitándole tener que introducir manualmente la dirección IP del dispositivo. [24]

Al haber usado el protocolo DHCP programado por Tiva y ver que funcionaba acorde a lo esperado, no se ha necesitado hacer cambios de esta parte. A continuación, se relata el funcionamiento de la parte de código que se encanrga de obtener la dirección de IP dinámica que aparecerá, junto con ostras instrucciones, al abrir la terminal de el programa. En este caso, se ha elegido usar el programa **Putty**, de licencia gratuita, que, además de cliente SSH y telnet, funciona como terminal a través de la UART para el programa. [30]

Para conseguir mostrar la IP en la sesión Putty, *enet_io* sigue una serie de pasos. En primer lugar, tras configurar la UART a 115200 baudios[2], muestra el mensaje «*Ethernet IO Example.*». Una vez hecho esto, y configurado el sistema para interrupciones periódicas, mirará si hay MAC programada y avisará de que está esperando una IP con el mensaje «*Waiting for IP.*» (esperando la IP=. Tras ello, llamará a la función *lwIPInit* para obtener la IP a partir del protocolo DHCP anteriormente mencionado.

Una vez dentro de la función, el programa usará funciones que se encuentran dentro de la librería *lwIP*, como por ejemplo la función *lwIPLocalIPAddrGet();*, que obtendrá la dirección IP actual. Posteriormente, la comparará con la dirección guardada. Si ha habido cambios, se pasará a mirar si:

- **La nueva dirección es igual a 0xffffffff**. En este caso, el programa esperará un enlace, es decir, que se conecte el microprocesador al enrutador a través del cable

[2]medida de la velocidad de transmisión de la UART

Ethernet. Imprimirá en la UART el mensaje «*Waiting for link.*» (esperando por un vínculo).

- **La nueva dirección es igual a 0**. Esto se suele dar tras el caso anterior y indica que el proceso DHCP está en marcha. Se verá en la terminal el mensaje «*Waiting for IP address.*» (esperando la dirección IP).
- **Se tenga una dirección diferente de las anteriores**. En este caso, ya se tiene una dirección IP que será mostrada en la terminal seguida del mensaje «*Open a browser and enter the IP address.*» (abra un navegador e introduzca la dirección IP).

Tras esto, la dirección IP se guardará para que pueda ser comparada cada vez que se llame a esta función. A continuación, en la figura 5.8 se muestra la UART tras el proceso de asignación de IP dinámica.

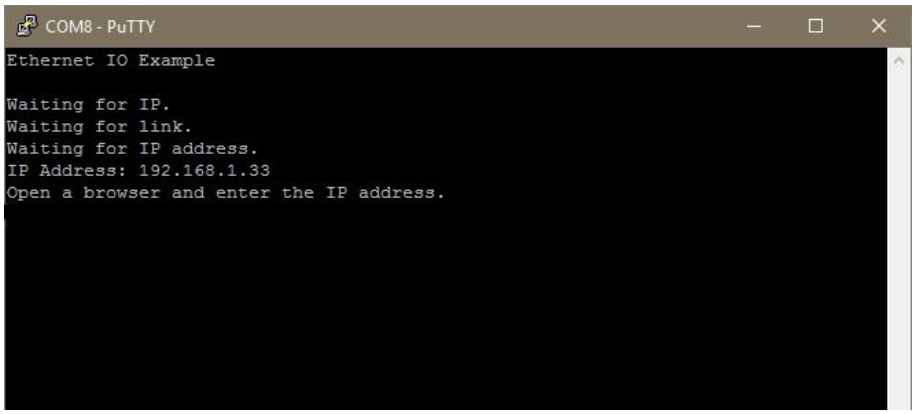


Figura 5.8: Terminal tras el proceso de asignación de IP mediante DHCP.

Además del programa de comunicación, las mediciones de humedad y temperatura y la obtención de direcciones IP, el ejemplo de *enet_io* cuenta con otros bloques de código como configuración de interrupciones, configuración de la UART y LED, o comunicación SSI. Dentro de las interrupciones, se ha modificado la del temporizador, como se podrá ver en el siguiente capítulo.

De las configuración de la UART y el LED que indica que todo está funcionando no se ha modificado nada en el programa principal, si bien se han tenido que eliminar algunas líneas en el archivo del programa de medición de sensores, puesto que estas se contradecían. Por ejemplo, en la UART, tenía que imprimirse al mismo tiempo el mensaje «*Waiting for IP*» y el mensaje del sensor «*Humidity SHT21*», dando lugar a un conjunto de caracteres ininteligible.

Capítulo 6

Programa de riego

En este capítulo se hablará del programa de riego propiamente dicho, de la adquisición de parámetros como la temperatura, la humedad o el tipo de ambiente elegido para realizar el riego en el programa oportuno.

Se ha decidido separar ese capítulo del anterior debido a su extensión, puesto que, al tratarse del grueso del trabajo, la explicación del programa completo en un sólo capítulo daría como resultado un capítulo demasiado extenso y complicado o bien una explicación incompleta. Sin embargo, cabe destacar que este apartado, llamado *Programa de riego* y el código asociado a él (presente en el Anexo 4[1]) pertencen al programa *enet_io* (como se ve en la figura 5.1 del capítulo anterior).

Otra razón para diferenciar ambos capítulos es su procedencia, puesto que, mientras todas las funciones del capítulo anterior eran adaptaciones del programa *enet_io*, en este se va a ver un código desarrollado desde cero y sin tomar como base ningún ejemplo proporcionado por *Tivaware*. Este código se ha situado dentro del bucle *while(1)* de la función principal *main*. De esta manera, es seguro que una vez dentro del bucle se realice el programa completo —parando la ejecución sólo en caso de de interrupción, como, por ejemplo, la petición al servidor— y, una vez finalizada la realización del programa, este vuelva al inicio para comenzar otra vez.

Para la realización de este código, se han dividido las tareas en bloques puesto que, aunque en un principio se pensó en utilizar funciones, al final se optó por esta manera para aumentar el rendimiento. Estos bloques funcionan uno tras otro dentro de un bucle infinito.

A pesar de que, en el código, se han programado seguidos; para una mejor visualización, en la figura 6.1 se ha decidido mostrar cada bloque como una función independiente que transcurre inmediatamente a continuación de la anterior. Sin embargo, cabe destacar que no son funciones, si no que sólo se ha hecho esto para poder analizar el extenso código en bloques más pequeños, que serán explicados con más detalle a continuación.

[1]Si bien a lo largo de este capítulo se exponen diferentes figuras con ejemplos de código, estas se han hecho con el fin de referenciar un apartado en concreto o esclarecer un punto. El código completo se encuentra en el Anexo 4.

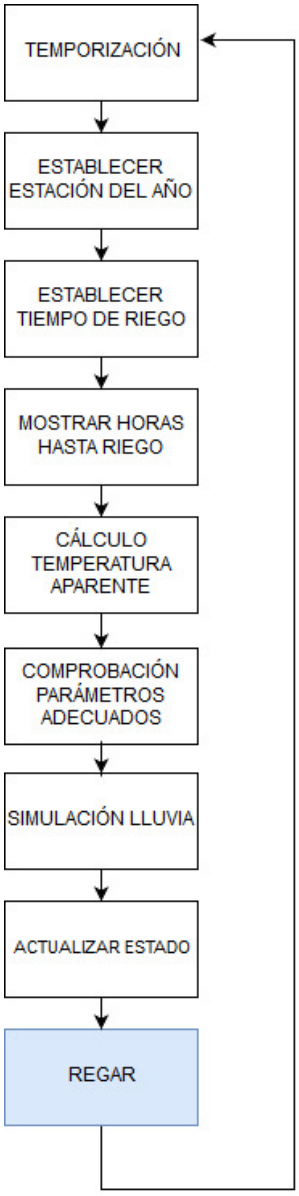


Figura 6.1: Diagrama de flujo del programa de riego

6.1. Temporización

El objetivo de este bloque es crear unas variables que nos permitan medir el tiempo. Esto es necesario para tres funciones principales: medir cada cuanto tiempo se riega, medir cuanto tiempo ha de estar la electroválvula de riego abierta y temporizar cada cuantas horas se ha de almacenar la temperatura para establecer en qué estación del año nos encontramos.

Para hacer la temporización se ha usado el timer —temporizador del microprocesador— que venía ajustado en el programa de ejemplo *enet_io* sobre el que se ha trabajado. Se

trata de un *Interval Timer*[2] con una interrupción cada 200 milisegundos (0,02 segundos).

Así, en lugar de programar otro timer, con lo que se restaría velocidad al microprocesador y se haría un uso innecesario de recursos, se ha aprovechado el que proporcionaba este programa *enet_io*. Para ello, se ha hecho uso de la variable «a» tipo *int*, que se incrementa cada vez que entra la interrupción, siempre y cuando se haya puesto a uno la variable «medir_tiempo» (puesto que, como se verá más adelante, puede que no interese medir el tiempo transcurrido en algunos instantes).

Luego, en el *main*, se ha utilizado esta variable «a» para comenzar el bloque de temporización, mostrado en la figura 6.2, que empezará a funcionar cuando se haya hecho la primera medida de temperatura y humedad. Cuando «a» llegue a 300[3] se reiniciará, y se incrementará en uno la variable «minutos», de tipo char (puesto que aquí no se necesitará que cuente más de 255).

Así mismo, cuando «minutos» llegue a 60, esta variable volverá a cero y se incrementará una unidad «horas», otra variable tipo *int*. Esto se debe a que, en algunos de los ambientes, se necesitará regar una vez cada 14 días (336 horas) y, con una variable tipo *char*, sólo se podría medir hasta 255 horas.

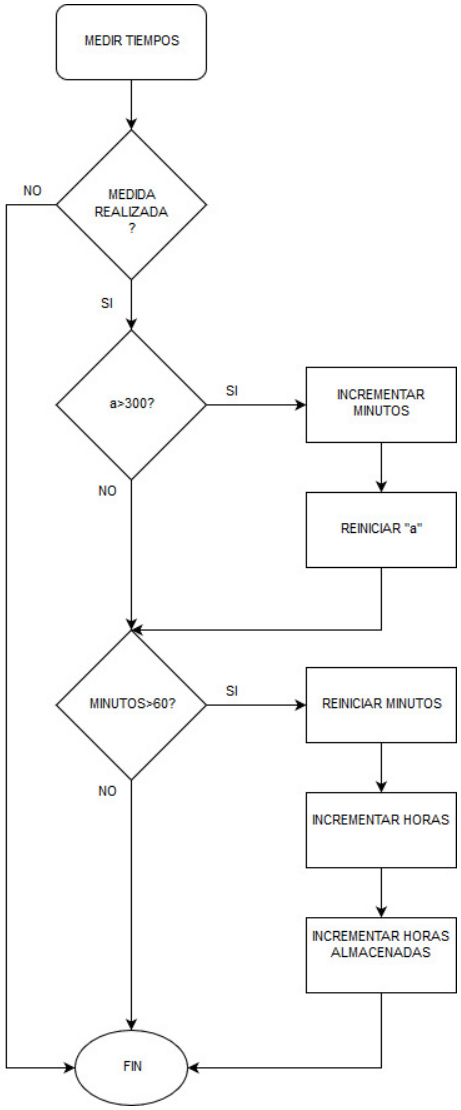


Figura 6.2: Diagrama de flujo de la parte de temporización

Por otro lado, al incrementar la variable «horas», se incrementará también «horas_alm». El uso de esta variable, así como su tipo, se explicarán en el siguiente punto: Establecer estación del año.

[2]Modo de temporización que cuenta ciclos y llama a una interrupción cuando se ha llegado al número de cuenta
[3]$60s/200ms$

6.2. Establecer estación del año

El objetivo principal de estas líneas de programa es medir la temperatura cada ocho horas y, con las últimas ocho muestras, asignar la estación teniendo en cuenta un clima continental. Para eso, se siguen los estos pasos:

En primer lugar, se deberá comprobar si se han medido ocho horas. Esto se hará mediante la variable tipo *char* «horas_alm», que se incrementa junto con la variable «horas» en la parte de temporización. La razón de hacer dos variables diferentes radica en que «horas_alm» sirve para almacenar el número de horas transcurridas independientemente de si ha habido riego o no (puesto que en caso de haberlo, «horas» se reiniciaría). Como, a diferencia de «horas», la variable que nos ocupa sólo contará hasta ocho, se ha decidido que sea tipo *char* y no *int*.

Si ha completado la cuenta, la variable se reinicia, se almacena la parte entera de la temperatura en un vector y se incrementa el valor de la posición para la vez siguiente con la variable «i» de tipo *char*. Cuando esta haya llegado a 7, mediante un bucle for, se comparan los ocho valores guardados con valores asignados y, si es una comparación positiva, se incrementa una variable. Así, si la temperatura es menor de 20 grados centígrados, se incrementará la variable «cuenta_temp_cold»; y si es mayor de 35 grados «cuenta_temp_hot» (ver Anexo D.2, correspondiente al código de esta sección).

Una vez hecho esto, sólo quedará asignar la estación. Para ello se ha decidido que si dos de las medidas superan los 35 grados o todas menos dos, los veinte, será verano. En caso contrario, será invierno. Esto se ha hecho así a fin de evitar la caída de temperaturas en las noches estivales.

Para elegir este criterio, se han evaluado los datos de la Agencia Estatal de Metereología (Aemet) de este último año, comparando las temperaturas medias del día y la noche en verano (julio) y en invierno (enero) y dándonos cuenta de que en verano, ni siquiera de noche, se alcanzaban mínimas de veinte grados excepto en casos muy puntuales, mientras que por el día se superaban fácilmente los 35. En invierno, sin embargo, rara vez, incluso en las horas más calurosas, se pasaba de los 21.

Además se ha llegado a la conclusión de que, haciéndolo de esta manera, se solventan mejor los cambios de estación. También, se evita que le afecten cambios de tiempo puntuales, —como las borrascas que hacen que baje la temperatura en verano— puesto que la medida se hará con la media de 8 datos guardados, es decir, tres días.

A continuación, en la figura 6.3, se muestra el esquema del este bloque[4]:

[4]La parte de color azul corresponde a la asignación de la estación.

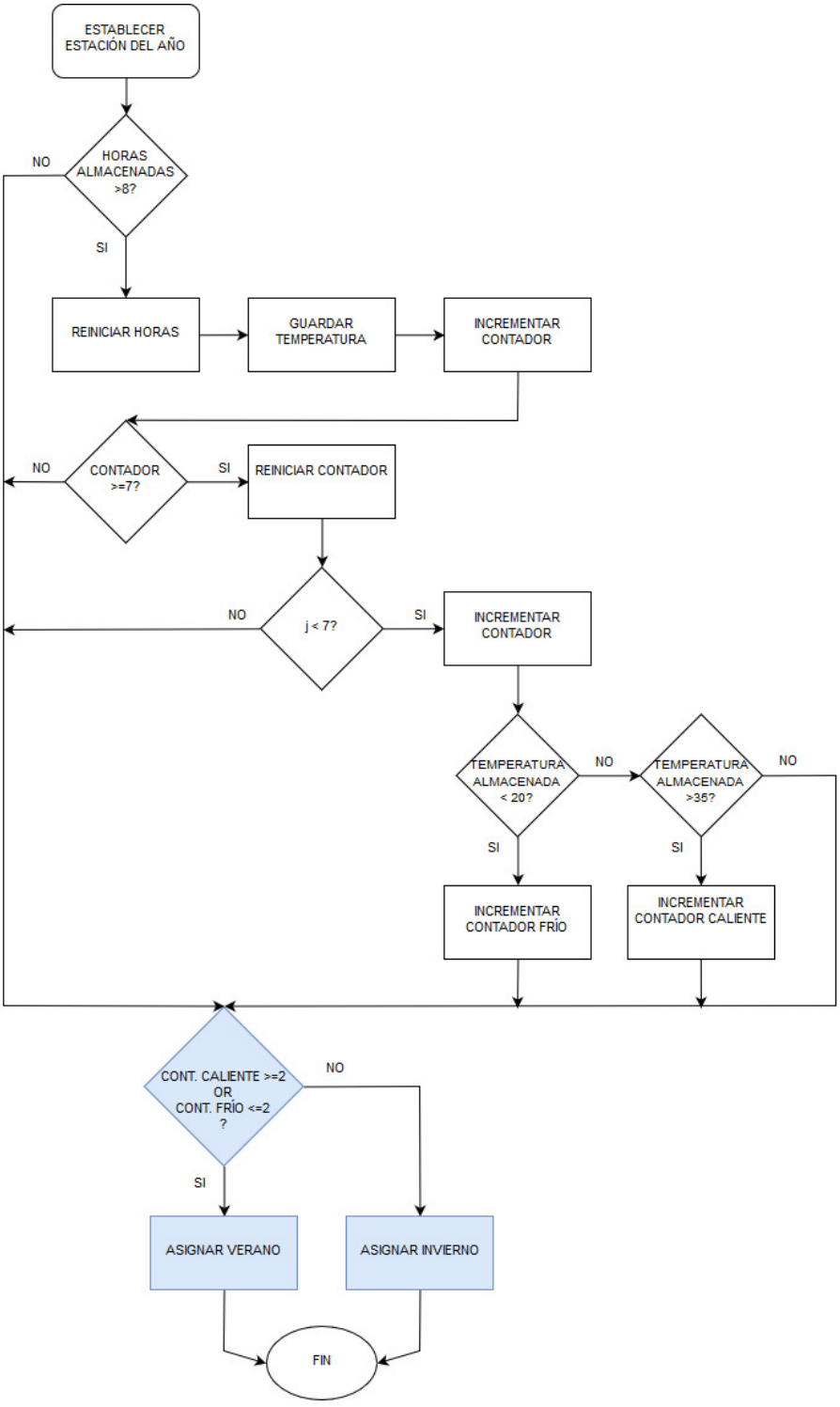


Figura 6.3: Diagrama de flujo de la parte de establecer estación del año

6.3. Establecer tiempo de riego

En este apartado, se hablará de cómo se asignan los tiempos para cada riego[5]. Para ello, primero se mirará el ambiente elegido, a continuación la estación del año (verano o invierno) y por último el tipo de terreno; y, a partir de los datos, se asignará un valor a la variable «tiempo_riego».

Como se puede ver en la figura 6.4, para implementarlo se usarán bucles if.

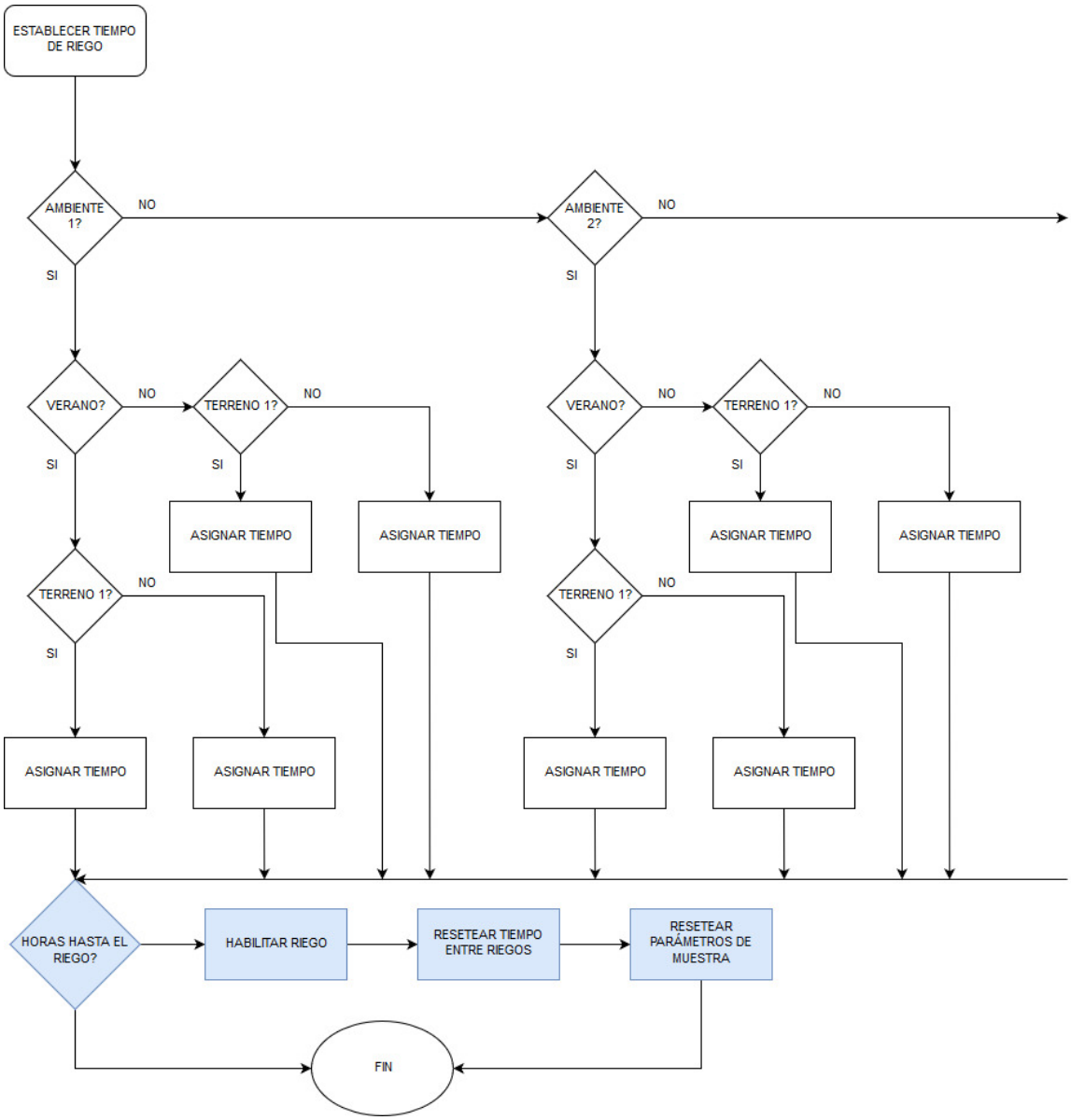


Figura 6.4: Diagrama de flujo de la parte de establecer el tiempo de riego

[5]Los intervalos de riego de cada ambiente se pueden ver en el Anexo 1.

De esta imagen destacan dos elementos. En primer lugar, se trata de un recorte, puesto que por problemas de tamaño no se ha podido incluir el diagrama de flujo completo (de hacerlo, sería imposible visualizarlo correctamente). Así pues, se ha elegido mostrar el proceso de asignación de los dos primeros ambientes, siendo el de los otros siete exactamente igual y cambiando sólo el número del ambiente.

El segundo elemento destacado, por otro lado, es la parte final del diagrama, en color azul, de la que aún no se ha hablado. Se trata de la comprobación de que ha pasado el tiempo suficiente para regar. Si al igualar las variables «tiempo_riego» y «horas», estas coinciden, se habilitará el riego mediante «ena_riego», se reiniciarán todas las variables de tiempo (excepto la que se usa para establecer la estación del año) y se reiniciarán también las variables que servirán para mostrar el tiempo restante en pantalla, lo que se verá en el próximo apartado.

6.4. Mostrar tiempo hasta próximo riego

Figura 6.5: Diagrama de flujo de la parte que muestra el tiempo restante

El objetivo de este bloque es calcular el tiempo restante hasta el siguiente riego para mostrarlo en la página principal de la web. Para ello se calcularán las horas hasta próximo riego restando el valor de la variable «horas» a «tiempo_riego» obteniendo así «horas_hasta_riego», que será la variable que se enviará a la web.

Cuando «horas_hasta_riego», valga 1, se calcularán los minutos restantes, siendo «min_hasta_riego» la diferencia entre 60 (por 60 minutos que tiene una hora) y «minutos» que serán los minutos transcurridos en esa hora. Al igual que con las horas, se enviará «min_hasta_riego» a la web y, para ello, se pondrá a uno «mostrar_min», que indica que se tiene que dejar de mostrar la variable que almacena las horas restantes para mostrar la que almacena los minutos.

Para hacer la simulación se ha elegido regar en intervalos de 3 y 5 minutos (dependiendo del terreno). Para mostrarlo correctamente, se ha utilizado la variable «tiempo_riego_min» que indica que se está programando minutos en lugar de horas. Esta se activa al establecer el tiempo de riego y, si al llegar a este bloque está a uno, activará «mostrar_min» y «min_hasta_riego» será la diferencia entre el tiempo de riego asignado (en minutos) y los minutos transcurridos.

6.5. Cálculo de la temperatura aparente

La temperatura aparente es la temperatura que siente una persona y que puede variar en función de la humedad a la que esté expuesta.[9]

Para calcularla en el programa, se esperará a que se haya realizado la primera medida —se ha puesto a uno la variable «measure_done» que ya se ha utilizado previamente en la temporización— y se calculará la temperatura aparente como:

TA= -9.93122+1.186145T+0.122310HR

Donde T será la medida de la temperatura y HR la medida del sensor de humedad.

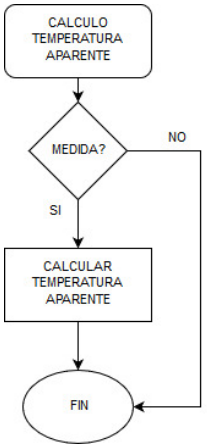


Figura 6.6: Diagrama de flujo de la parte de cálculo de la temperatura aparente

6.6. Comprobar si los parámetros son adecuados

Para evitar regar que las plantas se hielen o sufran un proceso similar a la cocción, independientemente del sistema de riego que se utilice (aspersión, goteo...), se deberá evitar el riego a partir de ciertas temperaturas. Esto se hará con un programa como el mostrado en la figura 6.7.

En esta se ve que, si se ha habilitado el riego con «ena_riego» —lo que significa que ya ha llegado el momento del riego— y se viene de un «estado = 0» —que quiere decir que se está en condiciones normales— se revisará la temperatura. Si es mayor de 40 grados, se cambiará «estado = 1» y pararemos de contar el tiempo. Esto se hace para que no transcurra el tiempo hasta el próximo riego, porque, como se verá más adelante, una vez se vuelva al rango de temperaturas aceptable, se volverá a regar. Así, podría darse el caso de realizar ese riego atrasado demasiado próximo al siguiente, algo que se evitará si se vuelve a contar el tiempo a partir de regar. Por otro lado, al cambiar al estado 1, en pantalla pasara a mostrarse «*Riego suspendido por altas temperaturas*».

La otra condición que podría retrasar el riego es si la temperatura es menor a 0 grados centígrados. De ser así, el estado pasaría a valer 2 y, al igual que en el caso anterior, se pararía la cuenta de tiempo para retrasar así el riego. En este caso, el mensaje a mostrar será «*Riego suspendido por riesgo de congelación*». El mensaje del estado 0 sería «*Esperando próximo riego*».

Por otro lado, si la temperatura está entre 0 y 40 grados, se pondrá a 1 la variable «act_riego» que activará el riego (antes se había habilitado, ahora se activará). La hume-

dad, al ser algo que interesaría conocer al instante, y no en el momento del riego, se ha tenido en cuenta en un bloque aparte, lo que se verá en el siguiente apartado.

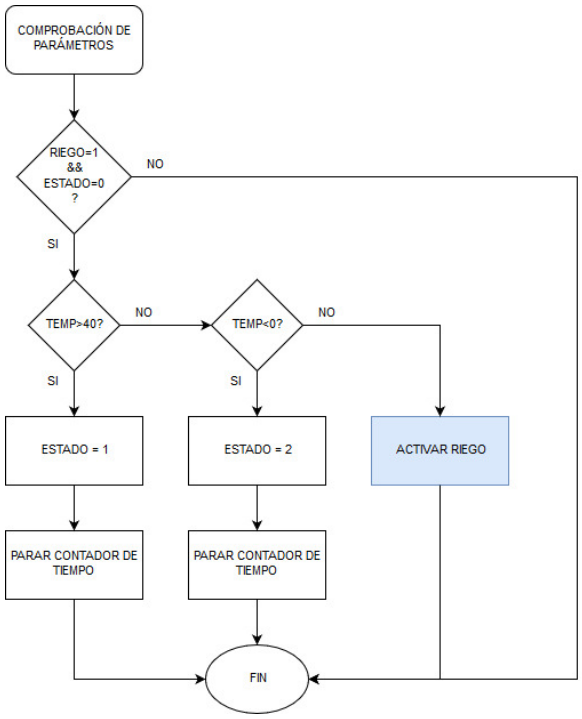


Figura 6.7: Diagrama de flujo de la parte de comprobar parámetros

6.7. Simulación de lluvia

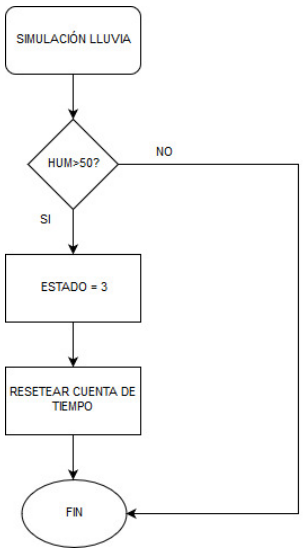


Figura 6.8: Diagrama de flujo de la parte que simula lluvia

Para simular ocasiones de lluvia, riegos manuales o fallos puntuales del sistema, se ha creado este bloque, representado en la figura 6.8.

Así pues, cuando la humedad supera el 50 %, se considera que la planta ya ha sido regada y se activará el estado 3 (riego cancelado por lluvias) mientras la humedad se mantenga en estos valores. También, para evitar que se riegue inmediatamente después, se pondrá a cero el tiempo transcurrido hasta el siguiente riego, con lo que se reiniciarán también las variables que muestran el tiempo restante por pantalla.

Este bloque se activará en cualquier momento, no sólo al habilitar el riego. Así, se consigue que la lluvia se considere un riego y el tiempo para el próximo se tome a partir de este.

6.8. Actualizar estado

Si previamente se había pospuesto el riego por muy altas o muy bajas temperaturas, estas líneas de código tienen como función si estas han vuelto a estar dentro del rango admitido. De ser así, como se muestra en la figura 6.9, se volverá al estado 0, y se activará el riego y se dará orden al contador de volver a medir el tiempo. No es necesario que se reinicie el temporizador, puesto que esto ya se había hecho al posponer el estado, tras lo cual se había parado el temporizador. Por tanto, sólo habrán transcurrido milisegundos, que resultan despreciable frente a las horas a medir.

Al igual que con la temperatura, si la humedad alcanza un valor dentro de lo permitido (menor al 50 %), el estado volverá a valer 0. Sin embargo, en este caso no se activará el riego inmediatamente después, puesto que, mientras la primera medida tenía como función evitar regar en las horas más calurosas en verano y durante las noches en invierno; en este caso un aumento de la humedad se considerará un riego y por lo tanto, el cambiar al estado 0 tan solo cambiará el mensaje de la pantalla de «*Riego suspendido por lluvias*» a «*Esperando próximo riego*», que son los correspondientes a los estados 3 y 0, respectivamente.

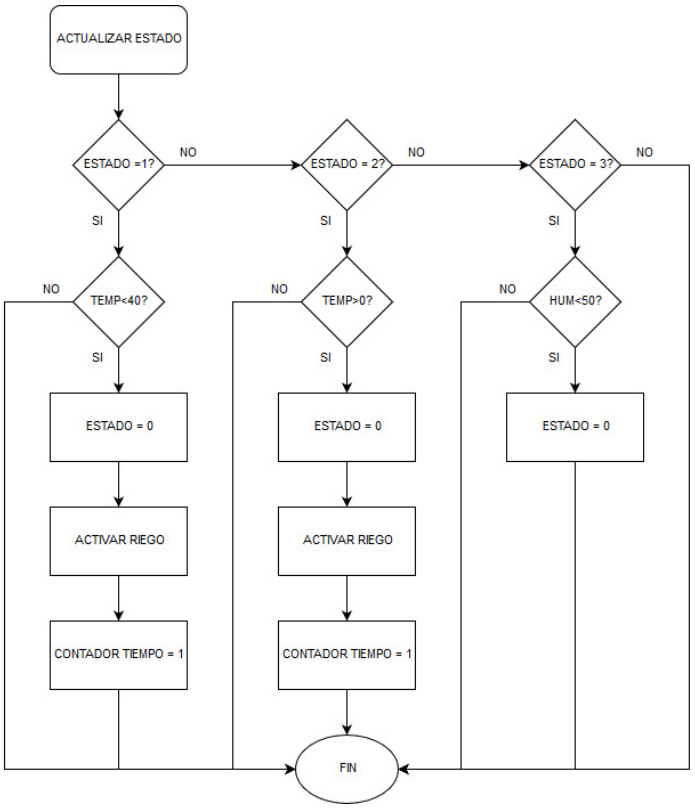


Figura 6.9: Diagrama de flujo de la parte de actualizar estado

6.9. Regar

Al llegar al final del bucle *while(1)*, dentro del *main*, y después de haber comprobado todos los parámetros y realizado todas las operaciones, llega el momento de regar. Si la variable «act_riego» vale 1, quiere decir que el riego está activado. Entonces, es el momento de poner un uno en el pin de salida.

Para esto, en primer lugar se deberá mirar el ambiente que se está regando, puesto que se ha elegido una manera diferente de regar el césped respecto a todo lo demás, lo que se verá en el capítulo 7. Una vez se sabe el ambiente, ahora sí, se activará la salida. Para ello se ha usado la función *GPIOPinWrite* que venía incluida en las funciones que proporcionaba Tiva en su librería *Tivaware*.

Para este ejemplo se ha elegido el pin PL0 para el riego normal (que será por goteo) y el pin PL1 para el riego del césped (riego por aspersión). Se tratan, ambos, de pines de propósito general que pueden ser configurados tanto de entrada como de salida. Así, la función de escritura quedaría como se muestra en la figura 6.10.

```
//Activar Pin que ponga a 1 la salida

    if (tipo6==1){ //Césped - Aspersor
        GPIOPinWrite(GPIO_PORTL_BASE, GPIO_PIN_0, GPIO_PIN_0);    //GPIO PL0
    }else{ // Riego por goteo
        GPIOPinWrite(GPIO_PORTL_BASE, GPIO_PIN_1, GPIO_PIN_1);   //GPIO PL1
    }
```

Figura 6.10: Código de selección de pin de salida

Tras activar el pin de salida, que abrirá a una electroválvula para dar comienzo al riego —lo que se explicará en el capítulo 7—, el siguiente paso a seguir es igualar la variable «estado» a 4. Así, el mensaje mostrado por pantalla en la página principal de la web será «*Regando*».A continuación se deshabilitará el riego poniendo la variable «ena_riego» a cero.

Al activar la variable «act_riego», en la interrupción del temporizador se empezará a incrementar una variable «a_riego» con la misma función que «a». En este bloque, una vez deshabilitado el riego, se comprobará si «a_riego» ha llegado a 300. De ser así se aumentará «minutos_riego» y se reiniciará. Esto permitirá contabilizar el tiempo que se tendrá la electroválvula abierta.

El siguiente paso será comprobar si han pasado los minutos que se debe estar regando. Sin embargo, como se ha decidido tener en cuenta las necesidades hídricas de cada tipo de planta, es necesario asignar a cada ambiente un tiempo diferente de riego[6]. Por lo tanto, el primer paso será comprobar el ambiente seleccionado, para posteriormente, comparar los «minutos_riego» con el tiempo asignado a esa planta.

[6]Los tiempos de riego de cada ambiente se muestran, junto con los intervalos, en el Anexo 1.

Si el tiempo de riego ha pasado, se deberá desactivar el riego, reiniciar las variables que contabilizan el tiempo de riego, volver el estado a cero y, por último, desactivar el pin PL0, que es lo que se puede ver en la figura 6.11 sombreado en rojo. Para realizar esto, se usará la función de Tiva *GPIOPinWrite(GPIO_PORTL_BASE, GPIO_PIN_0, 0x0);*. Por otro lado, si el ambiente seleccionado es césped, el pin que se deberá desactivar será el PL1, lo que se hará con la función *GPIOPinWrite(GPIO_PORTL_BASE, GPIO_PIN_1, 0x0);*

A continuación, se mostrará el diagrama de flujo que, al igual que en el capítulo *6.3. Establecer tiempo de riego*, ha sido necesario recortar por el tamaño total de este, que hacía imposible su correcta visualización. Sin embargo, el resto de ambientes se comportarían del mismo modo, siempre teniendo en cuenta el caso especial del ambiente 6, césped, que activa el pin PL1 en lugar del pin PL0. Para poder hacer un cuadro común y solventar que no aparezca este ambiente, se ha decidido no especificar en el diagrama el pin que se pone a 1, habiendo sido previamente explicado en párrafos anteriores.

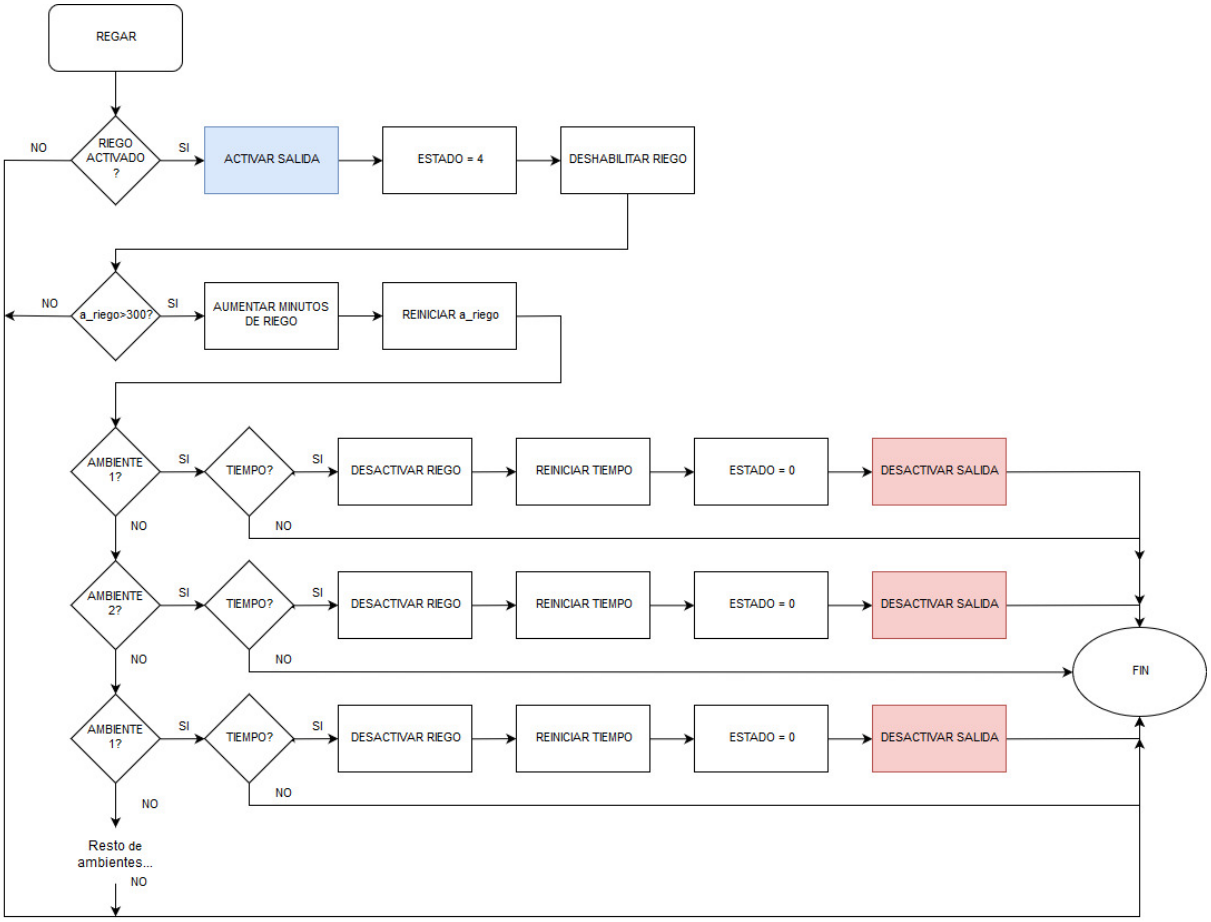


Figura 6.11: Diagrama de flujo de la parte de actualizar estado

6.10. Resultados

A continuación, se pueden ver unas capturas del funcionamiento del programa sometido a diferentes estímulos. Ya se mostró en la figura 4.12 —en el capítulo 4—- el estado normal de la página principal, que, con la variable «estado=0» muestra el mensaje «*Esperando próximo riego*». A continuación se mostrarán los ejemplos de los estados 1, 3 y 4, respectivamente en las figuras 6.12, 6.13 y 6.14
.

Figura 6.12: Ejemplo de riego suspendido por altas temperaturas

Figura 6.13: Ejemplo de riego suspendido por lluvias

Figura 6.14: Ejemplo de programa regando

En el estado 2, el cual no se ha conseguido reproducir llegando a los cero grados centígrados, el mensaje mostrado sería «*Riego suspendido por riesgo de congelación*».

Capítulo 7

Sistema de riego

En los capítulos anteriores se ha analizado el programa de riego, desde su interfaz web hasta el código que hace posible el funcionamiento del riego en los momentos adecuados. A continuación, se propone un ejemplo de sistema de riego que contenga todos los elementos necesarios para llevar a cabo la práctica de la maneras más eficiente y sostenible posible.

A pesar de que se ha decidido completar el proyecto con las siguientes herramientas, se podría haber elegido otras en función del presupuesto disponible.

7.1. Encapsulado y fuente de alimentación

En primer lugar,se hablará del encapsulado del microcontrolador y el pack de sensores. Se tratará de una caja con aberturas para los cables —de alimentación y Ethernet y dos cables salientes para las conexiones de los pines que darán la orden de riego de la electroválvula—, de medidas 138 x 56 x 27 mm (suponiendo el pack de sensores acoplado al microprocesador). Para ello se ha elegido una caja de medidas 155 x 95 x 57 mm — la más cercana a las medidas del conjunto—. Esta, además, da cierto margen en los laterales (17 mm) que permitirá introducir los cables por los orificios y proteger correctamente los cabezales de estos.

Por otro lado, como se puede ver en la figura 7.1, la caja no trae de serie los agujeros para los cables de alimentación y Ethernet ni los de los pines, por lo que será necesario mecanizarla. Por esta razón se ha elegido una caja de fácil mecanizado. Además de esto, interesa que la caja tenga un *Grado de protección IP* en tono a 63. El Grado de protección IP es una norma que ayuda a identificar cuán protegidos están los objetos frente al polvo (primer dígito) y el agua (segundo dígito) [58].

Figura 7.1: Ejemplo de encapsulado para las tarjetas de sensores ©Electrónica Embajadores. Imagen obtenida de [7]

Así pues, dadas las condiciones del proyecto, se ha considerado que sería adecuado tener una protección 6 frente al polvo (la más fuerte, el polvo no puede entrar de ninguna manera, evitando así que interfiera en el funcionamiento del microcontrolador) y una protección 3 frente al agua. Esta, que protege frente a agua nebulizada, permitirá medir la humedad evitando que el agua entre a la caja.

Respecto a la fuente de alimentación, se ha elegido una fuente de energía renovable, la energía solar. La razón ha sido porque esta es mucho menos contaminante y es más fácilmente adaptable a un jardín o invernadero. Sin embargo, la conexión final deberá elegirla cada usuario teniendo en cuenta las condiciones del lugar de la instalación.

Para poder aprovechar la energía solar se ha elegido un kit que contiene un panel fotovoltaico, un regulador y una batería, como el que se puede encontrar en [25]. En este caso, teniendo en cuenta que se va a tener un consumo pequeño, se ha elegido un panel que proporciona una energía de 400Wh/día. Anexo a este, se tiene un regulador de onda PWM. Se ha elegido este, y no uno MPPT, porque se pretende tener un consumo bajo, de corriente moderada, para lo cual el regulador PWM ofrecía un servicio correcto abaratando el coste respecto al MPPT [26].

Debido a que el panel solar no puede estar trabajando en todo momento a las mismas condiciones —noches, días nublados o con lluvia, diferencia de horas de sol en verano o invierno...— se ha completado el kit con una batería de 12 voltios que permitirá tener energía en todo momento. Así, como se muestra en la figura 7.2, se puede alimentar el resto de elementos bien directamente del regulador[1], bien de la batería en caso de que el panel no proporcione energía en ese momento.

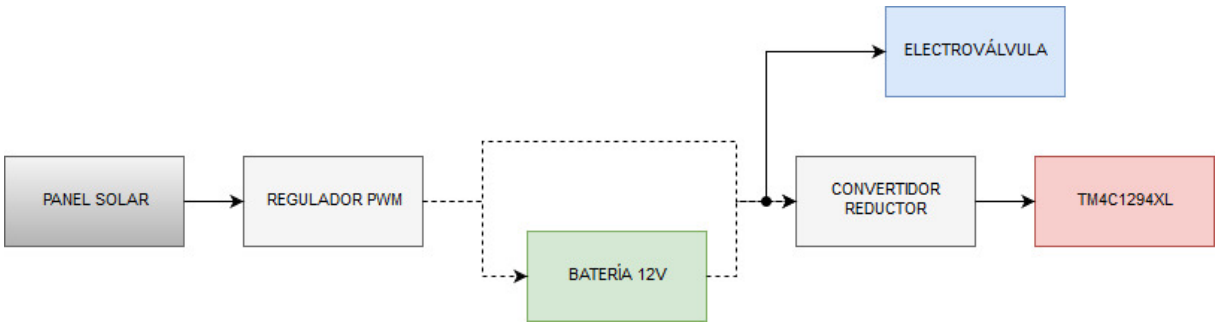


Figura 7.2: Diagrama de conexiones de la fuente de alimentación

Por otro lado, como muestra la figura anterior, se puede alimentar la electroválvula a 12 V, pero el microcontrolador requiere una tensión menor. Por eso, será necesario acoplarle un reductor de tensión a 5 o 3,3 V (funciona en ambos valores y la mayoría de reductores operan entre estos). También hay que fijarse en la corriente; si bien el regulador PWM permite operar hasta con 10 A de corriente máxima, no se puede utilizar una corriente

[1]Este regulador PWM puede trabajar con 12V o con 24V. Para alimentar la electroválvula se ha elegido el modo que ofrece 12V.

tan alta en los elementos conectados, por lo que se deberá procurar que el regulador de una corriente menor o poner elementos resistivos.

7.2. Válvulas

Para permitir el paso de agua desde el depósito o pozo en el que se encuentre hasta el cabezal de riego correspondiente, se necesitará una válvula que regule el paso. Para ello, se ha elegido una válvula de 4 vías y 3 posiciones, como la mostrada en las figuras 7.3 y 7.4.

Figura 7.3: Válvula 4/3 ©DISUMTEC. Imagen obtenida de [4]

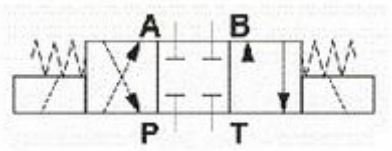


Figura 7.4: Esquema de la válvula de 4/3 ©DISUMTEC. Imagen obtenida de [4]

Como se puede ver en esta última figura, correspondiente al esquema de la válvula, esta tiene 3 posiciones. La posición del centro es la que adoptará la válvula normalmente: cerrada. Para cambiar a cualquiera de las dos posiciones adyacentes, la válvula dispone de dos accionadores —correspondientes a las dos posiciones, uno a cada lado de la válvula— electromagnéticos. Esto quiere decir que la válvula cambiará de posición al recibir una pequeña corriente en esos accionadores. Esta corriente, a su vez, vendrá dada por los pines de la válvula, que se podrán a uno cuando llegue el momento de regar.

La válvula se mantendrá en su posición correspondiente —dependiendo si se ha elegido césped y aspersores o cualquiera de los otros ambientes y riego por goteo— mientras el pin de riego esté activado. Cuando este vuelva a cero, dejará de pasar corriente y los muelles de retorno devolverán la válvula a su posición de cerrado.

Por otro lado, la válvula posee 4 vías, pero sólo se utilizarán 2, puesto que el agua será expulsado por los aspersores o por el sistema de riego por goteo y no se necesitará un canal de retorno para cerrar el circuito hidráulico.

Además de esta válvula, también se necesitará una válvula reductora de presión. Esto se debe a que, en el apartado siguiente, cuando se elija una bomba, se deberá escoger

una capaz de aportar la presión suficiente para el funcionamiento de los aspersores; sin embargo, demasiada presión podría dañar el sistema de riego por goteo. Por ello, en la tubería que sale de la válvula 4/3 se deberá colocar una reductora para asegurar que la presión del agua es correcta para el trabajo.

Las válvulas descritas en este apartado han sido pensadas para el caso de que el usuario prefiera hacer uso de un sistema flexible en el que pueda cambiar de césped a cualquiera de los otros ambientes, o viceversa. En caso de que se decida utilizar el sistema para un uso concreto, sería suficiente con una electroválvula normal, de dos posiciones (abierta y cerrada) y dos vías (para que conduzca el agua en una sola dirección), como la mostrada en la figura 7.5[2]. Para su funcionamiento, se debería conectar el accionador al pin correspondiente en función del tipo de riego (aspersión o goteo) deseado. En este caso tampoco sería necesario la válvula reductora de presión.

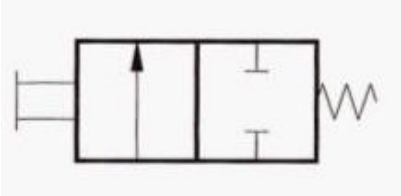


Figura 7.5: Esquema de la válvula de 2/2

7.3. Bombas

La bomba es el dispositivo que se encarga de extraer el agua del pozo o depósito que se tenga instalado y conducirla por las tuberías hasta la válvula que permitirá el paso del agua.

Según lass necesidades del usuario, se podrán elegir bombas con diferentes características [28]. En función de estas características se pueden hacer las siguientes clasificaciones de las bombas:

- **Electrobombas/ motobombas:** La distinción se hace por el tipo de motor que estas tienen. Mientras las electrobombas funcionan enchufadas a la red eléctrica (220 voltios de corriente alterna), las motobombas requieren de combustible, generalmente gasóleo, para funcionar. Estas segundas son más útiles cuando no se dispone de una instalación eléctrica en el lugar destinado al riego.

- **Bombas sumergibles/ no sumergibles:** Aquí, la diferencia se halla en la profundidad de la que se quiere extraer el agua. Para depósitos en superficies o pozos de menos de ocho metros de profundidad, se utilizarán las no sumergibles, mientras que para pozos mayores, las sumergibles.

[2]En este caso, en lugar del accionamiento electrónico que se necesitaría poner, viene representado un accionamiento manual.

- **Bombas horizontales/ verticales:** Mientras las bombas verticales aportan una mayor presión, las horizontales disminuyen esta un poco a favor de suministrar un caudal elevado.
- **Bombas de agua limpia/ bombas de agua sucia:** En referencia a si se trata de un pozo de agua limpia o este también tiene barro que ha de ser filtrado.

Además, se pueden clasificar por caudal, potencia en vatios y presión suministrada. Por otro lado, cabe destacar que existen bombas para otro tipo de fluidos, como aceite, que no se considerarán al no ser aptas para el riego. [18]

Para llevar a cabo la puesta en práctica de este proyecto se ha elegido una bomba con las siguientes características:

- **Electrobomba:** De este modo, como se explicó en el apartado 2.4., se consigue un mayor ahorro energético. Sin embargo, no se ha podido conseguir un ahorro total conectándolo a una placa solar debido a que la electrobomba consume mucha más energía y necesita, al menos, 220 voltios de corriente alterna. Por esto, se considerará una solución más óptima y ajustable al presupuesto conectar la bomba de agua a la red eléctrica.
- **Bomba no sumergible:** Considerando el área de trabajo como invernaderos, superficies comerciales o jardines privados, se ha supuesto que la toma de agua se hará mediante un depósito.
- **Bomba horizontal:** Así mismo, las bombas más recomendadas para el riego son las horizontales.
- **Bomba de agua limpia:** Al considerar un depósito, no se cree que se deba hacer uso de una bomba especializada en barro y fango.

Dentro de las diferentes bombas del mercado, para el caso de estudio interesaría una que alcanzase altas presiones, para así poder hacer funcionar los aspersores de riego. Mientras que el riego por goteo no necesita demasiada presión (la suficiente para recorrer las tuberías), un aspersor necesita mucha más. Así, considerando la posibilidad de que se puedan usar ambos casos, se ha elegido una bomba que suministre suficiente presión para poner en funcionamiento el aspersor, reduciendo esta presión si fuese necesario —un cambio de ambiente— mediante una válvula reductora.

También sería interesante utilizar bombas de accionamiento automático, para, de esta manera, encenderlas solamente si son necesarias, suponiendo un cuantificable ahorro. Usualmente este accionamiento se consigue mediante reguladores de presión y se añaden como un complemento a la bomba. [1]

Sin embargo, la elección de la bomba depende completamente de la localización de la zona de riego y del criterio propio del usuario.

7.4. Tipos de riego

Existen numerosos tipos de riego: por aspersión, por microaspersión, por goteo, por nebulización, etc. Este apartado se centrará en los riegos por aspersión y los riegos por goteo, que son los que se proponen para este trabajo, si bien, conectándolo al pin correspondiente, el usuario puede elegir los riegos que más le convengan[3].

Riego por aspersión

Hablamos de este riego si se utilizan aspersores para distribuir el agua. Este método se ha elegido para el ambiente césped debido a que consume más que el riego por goteo (una de las razones por las que no se ha extendido su uso a los demás ambientes) y, principalmente, porque permite un riego homogéneo de toda la superficie cubierta, siendo esta superficie mucho mayor que la de riego por goteo.

En las figuras 7.6(a), 7.6(b) y 7.6(c) se pueden ver 3 tipos diferentes de aspersores. Estos se pueden clasificar de diversas maneras: por caudal, por material de fabricación, por posición (sobre el césped, con piqueta), por movimiento (fijo, oscilante) o por área y radio de riego (por ejemplo, rectangular de 10 metros o circular en un radio de 8 metros). [17]

(a) Aspersor oscilante (b) Aspersor con base (c) Aspersor con piqueta

Figura 7.6: Diferentes ejemplos de aspersores ©Leroy Merlin. Imagen obtenida de [17]

[3]Siempre teniendo en cuenta que los tiempos de duración del riego del programa han sido pensados para riego por goteo y aspersión (en el caso del césped).

Riego por goteo

> «De forma muy general, se puede definir el Riego por Goteo como Riego Localizado. El riego por goteo o riego gota a gota es un método de irrigación que permite una óptima aplicación de agua y abonos en los sistemas agrícolas de las zonas áridas. El agua aplicada se infiltra en el suelo irrigando directamente la zona de influencia radicular a través de un sistema de tuberías y emisores.»

(Novedades Agrícolas, 2016: [29])

Para todos los ambientes, exceptuando el césped, se ha elegido el riego por goteo. Este es el más eficiente, y que mayor ahorro de agua proporciona. Además, profundiza en las tierra llegando a empaparla bien, lo que es especialmente útil en el caso de algunas plantas, como los árboles.

También es totalmente aconsejable este sistema de riego si las plantas se encuentran en una superficie comercial o se quiere regar plantas de interior, puesto que un aspersor no tendría precisión suficiente para regar plantas en maceteros. Además, de tratarse de una superficie comercial y activarse el riego en horario de apertura al cliente, el riego por goteo es el menos agresivo, mientras que el riego por aspersión podría resultar catastrófico.

A continuación, se muestra una imagen de ejemplo del riego por goteo, en la figura 7.7.

Figura 7.7: Ejemplo de riego por goteo ©Novedades Agrícolas. Imagen obtenida de [29]

7.5. Solución

Por último, se adjunta en la figura 7.8, el esquema de conexión de todos los elementos propuestos en los apartados anteriores, tales como bomba, electroválvula y cabezales de riego.

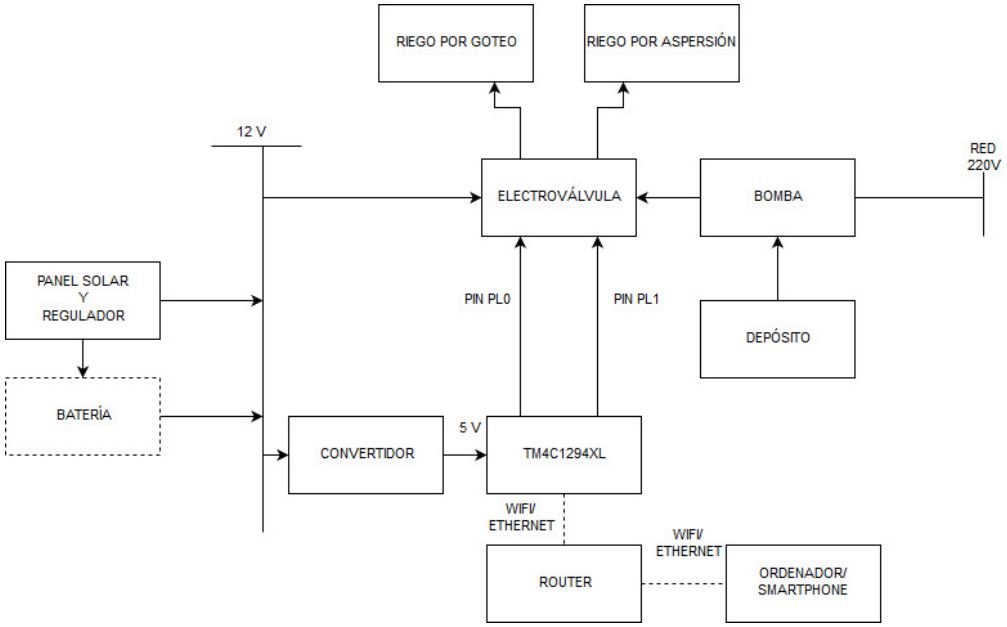


Figura 7.8: Esquema de conexión de los elementos del sistema de riego

Capítulo 8

Conclusiones

A continuación, se presenta una disertación acerca de los resultados finales del proyecto y la consecución de objetivos que se detallaba en el capítulo1. Así mismo, se proponen una serie de posibles mejoras futuras que se podrán implementar teniendo este proyecto como base y haciendo uso del código utilizado.

8.1. Conclusión

Para realizar la aplicación que ocupa este proyecto, se ha debido de seguir los siguientes objetivos que, ordenados por orden de relevancia han sido:

- Crear un programa de riego que tenga en cuenta la temperatura, humedad, y parámetros seleccionados para asignar los tiempos de riego y calcular la temperatura aparente. Además, este programa deberá modificar el estado del riego en cuanto se notifique una «perturbación» y actualizar el tiempo restante en horas. Cuando sólo quede una hora, deberá mostrarlo en minutos.
- Crear una interfaz web tipo formulario que permitiese al usuario registrarse y guardar sus datos o acceder en caso de estar registrado.
- Ampliar la interfaz web de tal modo que permita al usuario interactuar con el microprocesador para seleccionar el número de ambientes y el tipo de terreno deseado.
- Sensorizar la humedad y temperatura de un área o región y almacenar los datos obtenidos.
- Mostrar en pantalla las medidas del sensor, el cálculo de la temperatura aparente, el estado del riego y el tiempo restante para el siguiente, así como los parámetros previamente seleccionados.

8.2. Desarrollos futuros

A continuación, se plantean ciertas mejoras que no han sido consideradas aptas para implementar en el trabajo bien por falta de conocimientos, bien por falta de recursos. Estas son:

- Conectar el microprocesador al router y transmitir información mediante WiFi en lugar de Ethernet. Para ello sería necesario añadir el módulo de expansión de Tiva, de coste $34.99(USD)[1] en la página oficial.
- Crear un botón que permita regar en el momento que el usuario quiera, independientemente de las condiciones climáticas. Esto no se llevó a cabo debido a que el servidor tardaba demasiado en responder y, a veces, caducaba el tiempo de espera.
- Permitir cambiar los ambientes sin necesidad de volver a registrarse. Al igual que el punto anterior, esta solución se implementó pero se desechó debido al tiempo de respuesta del servidor y a la falta de conocimientos para encontrar una solución óptima, si bien se intentaron varias con Javascript.
- Crear un programa con varios ambientes interconectados entre sí. Para ello, se dispondría de varias microprocesadores con su respectiva tarjeta de sensores (tantos como ambientes se requieran). Cada microprocesador mediría los parámetros de su ambiente, y el riego sería gestionado por una tarjeta maestra. La solución que se propone es mostrar cada ambiente por separado en la página principal, como se muestra en la figura 8.1.

  Esta solución se implementó pero no se llevó a cabo al no tener demasiado claro el protocolo de comunicación máquina-máquina mediante telnet. En caso de querer llevarse a cabo, se recomienda buscar el ejemplo *enet_s2e* de Tiva.

Figura 8.1: Solución propuesta para la página principal teniendo varios ambientes

[1]Más gastos de envío.

Para el formulario se usaría un «checkbox» para seleccionar el ambiente y una lista desplegable para decidir qué tipo de planta querría el usuaroio en este (ver figura 8.2).

Figura 8.2: Solución propuesta para el formulario de selección de varios ambientes

Anexos

Anexos A

Anexo I: Tablas de riego

A continuación, en la tabla A.1, se muestran los intervalos de tiempo, en horas, tras los cuales se regará cada planta.

			Suelo árido		**Suelo húmedo**	
			Verano	**Invierno**	**Verano**	**Invierno**
TIPOS DE PLANTAS	**Exterior**	**Árboles**	168	360	360	720
		Palmeras	48	96	96	360
		Huerto	48	168	96	240
		Flores	12	72	24	120
		Cactus	360	-	720	-
		Césped	8	72	12	120
	Interior	**Plantas con flor**	24	48	48	168
		Plantas sin flor	48	96	72	168
		Orquídeas	72	120	120	168

Tabla A.1: Tiempos entre riegos para cada tipo de planta en diferentes terrenos y estaciones del año

Estos tiempos de riego se refieren a una media de los datos recogidos a través de diversas fuentes entre las que se encuentran varias páginas de Internet, libros sobre riego y personas consultadas. En estos, no se da una medida concreta de tiempo, si no que se dice que, aproximadamente, se debe regar cada cierto tiempo. Si bien se ha procurado tomar medidas estándar propicias para un clima continental de veranos secos, puede que una anomalía climatológica o una situación en climas más húmedos haga que estos riegos no sean suficientes o por el contrario se sobrepasen en su función.

También se ha pensado en las plantas en un estado de maduración óptimo. Si se trata de árboles pequeños, que necesitan más agua, se les deberá poner un programa más intensivo como por ejemplo «Huerto». Así mismo, los árboles frutales también se han considerado parte del huerto, al necesitar, por su condición, más agua que los árboles ornamentales.

La siguiente tabla nos muestra el tiempo que está activado cada riego (ver tabla A.2). Si bien el ambiente «Césped» presenta sólo 5 minutos de riego, se ha de señalar que el uso de aspersores conlleva que la cantidad de agua repartida por minuto sea mayor.

			Minutos
TIPOS DE PLANTAS	**Exterior**	**Árboles**	10
		Palmeras	5
		Huerto	10
		Flores	7
		Cactus	5
		Césped	5
	Interior	**Plantas con flor**	7
		Plantas sin flor	5
		Orquídeas	5

Tabla A.2: Tiempos de riego

Para la simulación se ha considerado más eficiente reducir los tiempos. Para ello, se ha elegido el ambiente «Césped», de exterior, y se han reducido sus tiempos entre riegos a 3 minutos en verano y terreno árido y 5 en terreno húmedo. Para invierno se han duplicado estos tiempos. También se ha reducido el tiempo en el que el aspersor estaría regando a 1 minuto, de manera que se pueda apreciar claramente el riego pero este suceda rápido para no entorpecer la simulación.

Nota: En invierno se ha considerado que con las lluvias es suficiente para regar los cactus, dado que estas son plantas que no necesitan demasiada agua. Por esta razón no aparece intervalo de tiempo en la tabla A.1.

Anexos B

Anexo II: Presupuesto

En el siguiente anexo se presupuestarán los costes que de este proyecto y se evaluará su entrada en el mercado. Se han dividido estos costes del proyecto en costes de material, costes de desarrollo y costes de fabricación; y se ha evaluado su presupuesto tanto para un número reducido de unidades como para un lote significativo, haciendo la pertinente evaluación entre ellas. Todos estos costes están en euros.

B.1. Coste de materiales

En la tabla B.1 se pueden ver incluidos todos los elementos físicos que son necesarios para el proyecto, así como algunos opcionales que, si bien han sido necesarios en el desarrollo del proyecto, el usuario puede no necesitarlos o encontrarse en posesión de algunos similares como son el router o el adaptador Ethernet - USB (para ordenadores sin puerto Ethernet). En estos, además, no se especifica marca ni modelo, puesto que a pesar de haber elegido unos concretos, el usuario puede preferir cualquier otro tipo.

Por otro lado, al final de la tabla se puede ver el ítem «Gastos de envío». Estos se refieren al microprocesador y a la tarjeta de sensores, proporcionados por TIVA. Todos los impuestos, aranceles y gastos de aduana van aquí incluidos. Para el router o el adaptador no se han incluido gastos de envío por la razón expuesta en el párrafo anterior.

ITEM	CONCEPTO	CANT.	IMPORTE	TOTAL
1	Microprocesador TIVA TM4C1294XL con cable USB y cable de red incluidos	1	16,61	16,61
2	Tarjeta de sensores SENSOR HUB BOOSTERPACK	1	41,53	41,53
3	Router (opcional)	1	29,00	29,00
4	Adaptador Ethernet - USB (opcional)	1	11,01	11,01
5	Caja IP 63 a medida	1	11,49	11,49
6	Gastos de envío	1	5,81	5,81

Total presupuesto	**115,45 euros**

Importa el presente presupuesto la cantidad de:
Ciento tres euros y noventa y seis céntimos.

Tabla B.1: Presupuesto de materiales

No se incluyen gastos de licencias de programas, documentación o código previo sobre el que se ha trabajado debido a que todos ellos eran de licencia gratuita — *open source*— y no han generado coste alguno.

B.2. Coste del desarrollo del proyecto

En esta sección se evaluarán los costes que ha supuesto el diseño y desarrollo del proyecto desde su idea inicial hasta el el producto final. Para ello se han dividido estos costes, puramente temporales, según la función a la que han sido dedicados. Así pues, como se puede ver en la tabla B.2, se tienen los apartados de: documentación y desarrollo de la idea inicial (donde se ha definido el problema), programación (tanto la creación de la web como el entendimiento y modificación del programa), prueba de los resultados y corrección de errores y, por último, tiempo empleado en la documentación de la memoria.

Para calcular el importe de las horas trabajadas, se ha recurrido al convenio de minas y a varios portales de Internet, como el conocido «Infojobs», de donde se ha extraído que el sueldo de un ingeniero junior está alrededor de los 2.000 euros al mes, lo que en una jornada completa supone 12,50 euros la hora.

ITEM	CONCEPTO	CANT.	IMPORTE	TOTAL
1	Horas de documentación y desarrollo de la idea	25	12,50	312,50
2	Horas de programación	282	12,50	3.525,00
3	Horas de prueba	19	12,50	237,00
4	Horas de memoria	176	12,50	2.200,00

Total presupuesto	**6.275,00 euros**

Importa el presente presupuesto la cantidad de:
Seis mil doscientos setenta y cinco euros.

Tabla B.2: Presupuesto del desarrollo del proyecto

B.3. Costes de fabricación

Como se debe mecanizar el encapsulado, se ha necesitado incluir el precio de un técnico de mecanizado. Aproximadamente serán quince minutos, pero se le pagará la hora completa. Recurriendo al método explicado en la sección anterior, se ha podido ver que el sueldo de un técnico de mecanizado ronda los 10,50 euros la hora.

Se ve en la siguiente tabla: (ver tabla B.3)

ITEM	CONCEPTO	CANT.	IMPORTE	TOTAL
1	Horas de técnico de mecanizado	1	10,50	10,50

Total presupuesto	**10,50 euros**

Importa el presente presupuesto la cantidad de:
Diez euros y cincuenta céntimos

Tabla B.3: Presupuesto de fabricación

B.4. Presupuesto total para una unidad

A continuación se muestra el presupuesto total para una unidad de SENSOR HUB BOOSTERPACK añadida a una unidad de tarjeta Tiva TM4C1294XL programada con el programa de riego desarrollado. En la tabla B.4 se puede ver cómo quedaría el presupuesto total, correspondiente a un importe de: **Seis mil cuatrocientos euros y noventa y cinco céntimos**.

ITEM	CONCEPTO	CANT.	IMPORTE	TOTAL
1	Microprocesador TIVA TM4C1294XL con cable USB y cable de red incluidos	1	16,61	16,61
2	Tarjeta de sensores SENSOR HUB BOOSTERPACK	1	41,53	41,53
3	Router (opcional)	1	29,00	29,00
4	Adaptador Ethernet - USB (opcional)	1	11,01	11,01
5	Caja IP 63 a medida	1	11,49	11,49
6	Gastos de envío	1	5,81	5,81
7	Horas totales de ingeniero	502	12,50	6.275,00
8	Horas totales de técnico	1	10,50	10,50

Total presupuesto	**6.400, 95 euros**

Importa el presente presupuesto la cantidad de:
Seis mil cuatrocientos euros y noventa y cinco céntimos

Tabla B.4: Presupuesto total para la fabricación de una unidad

Se han desglosado los costes de material para poder hacer una fácil comparativa con el presupuesto de fabricación por lotes, que se analizará en detalle en el siguiente apartado.

B.5. Presupuesto para una unidad con fabricación por lotes

En las secciones anteriores se ha podido ver como el coste de horas totales de desarrollo del proyecto superaban con creces los costes de material y fabricación. En este apartado se pretende mostrar como, si se fabrica un lote grande, estos y otros costes disminuirían hasta el punto de hacer un coste por unidad más moderado. Para ello se va a tomar como ejemplo un lote de mil unidades

En primer lugar se tienen los ya mencionados costes de desarrollo, aquí nombrados como horas totales de ingeniero. Estos comprenden 6.275,00 euros, pero, al usar el mismo programa para todas las unidades, se podrán dividir estos costes entre todas las unidades del lote, disminuyendo así hasta 6,28 euros.

Además, el coste de las horas de técnico también se ve reducido, puesto que antes se consideraba que podría hacer más de una placa en una hora pero se le abonaría el importe de toda la hora. Ahora, considerando que puede hacer unas cinco cajas por hora, el coste total se ve reducido en 8,40 euros.

También se ve una disminución, aunque menor, debido a que los gastos de envío se tornan prácticamente nulos (se considerarán nulos en el presupuesto) y existe una rebaja en el precio de las cajas de encapsulado. Así, el presupuesto para una unidad de un lote de mil comprende un total de: **Ciento diecisiete euros y trece céntimos**. Ver tabla B.5.

ITEM	CONCEPTO	CANT.	IMPORTE	TOTAL
1	Microprocesador TIVA TM4C1294XL con cable USB y cable de red incluidos	1	16,61	16,61
2	Tarjeta de sensores SENSOR HUB BOOSTERPACK	1	41,53	41,53
3	Router (opcional)	1	29,00	29,00
4	Adaptador Ethernet - USB (opcional)	1	11,01	11,01
5	Caja IP 63 a medida	1	10,6	10,6
6	Gastos de envío	0	5,81	0
7	Horas totales de ingeniero	0,5	12,50	6,28
8	Horas totales de técnico	0,2	10,50	2,10

Total presupuesto	**117, 13 euros**

Importa el presente presupuesto la cantidad de:
Ciento diecisiete euros y trece céntimos.

Tabla B.5: Presupuesto para una unidad con fabricación por lotes de mil unidades

Nota: Nótese que no se ha añadido a estos costes los costes por revisión de lotes en la fabricación.

B.6. Impacto económico en el mercado

Si bien fabricar en lotes de 1000 unidades puede resultar arriesgado para una primera toma de contacto con el cliente, se ha visto que la fabricación por lotes, incluso más pequeños, reduce considerablemente el costo final al dividir los costos de desarrollo. Así pues, incluso con un lote diez veces más pequeño (100 unidades) el presupuesto final resulta de 173,60 euros.

Analizando los competidores en mercado, se podrá ver que el más cercano, tanto por precio como por prestaciones el el modelo Fliwer que, a pesar de no presentar interfaz web, se vende a 149,00 euros. Si, además, se elige conectarlo mediante WiFi o 3G, el precio será de 299,00 euros. Además de este, se pueden ver otros ejemplos como los de Gardena o los de Idis, aunque estos no son tan parecidos. Gardena vende su sistema Smart por más de 500 euros e Idis, dependiendo de la función deseada, por más de 3.000 euros.

Así, con un costo de 173, 60 euros, se puede aplicar un amplio margen de beneficio sin perder competitividad en el mercado, siendo este un proyecto rentable. Además, gracias a su precio bajo, no sólo invertirán grandes compañías o superficies comerciales, sino también particulares interesados en tener un de riego inteligente en su jardín.

Nota: Cabe destacar que en el presupuesto no se ha incluido el sistema de riego con bombas, válvulas y aspersores. Esto se debe a que es algo muy específico de cada jardín y de las necesidades de cada usuario, y se pretende llegar a un mercado mayor ofreciendo ambos servicios por separados para que, en caso de que un posible cliente ya disponga de un sistema de riego previo, pueda automatizarlo con este sistema.

Anexos C

Anexo III: Código HTML

En el siguiente anexo se presenta el código correspondiente a la realización de la página web, así como las funciones JavaScript que permiten su correcto funcionamiento. El código CSS, por tratarse de un aspecto meramente estético, y no influir en el desarrollo de la página, se ha decidido no incluirlo.

El fin de introducir este anexo, así como el siguiente, es el facilitar la comprensión del desarrollo del trabajo a todo aquel que lo lea, así como proporcionar el código en caso de que se deseen hacer modificaciones.

Todo el código que aquí se presenta se ha elaborado gracias a la página *W3 Schools* y al foro de programación *stackoverflow.com*, basándose en los consejos y ejemplos allí presentes. Por otro lado, las páginas de «*Unirse*» y «*Registro*» se han realizado siguiendo diversos tutoriales en línea.

C.1. Inicio

```
1
<!DOCTYPE html>
3 <html lang="es">
<head>
5   <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js">
    </script>
7   <script src="javascript.js"language="JavaScript1.2" charset="utf-8"></script>

9   <meta charset="iso-8559-1" />
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
11  <meta name="descritpion" content="Control de riegos" />
    <meta name="keywords" content="riegos, ambientes, control" />
13
    <title>Control de riegos por ambientes</title>
15  <link rel="stylesheet" href="mihojadeestilos_ind.css" />

17 </head>

19 <body>
```

```
<a id="escudo" target="_" href="http://www.uc3m.es">
    <img src="uc3m.png" alt="UC3M">
</a>

  <div id="wrapper">
      <form method=post autocomplete="on">
        <h2>CONTROL DE RIEGOS POR AMBIENTES</h2>
        <p class="to_login">
          <a id="button1" href="login.html">Unirse</a>
        </p>
        <p class="to_register">
          <a id="button2" href="register.html">Registrarse</a>
        </p>
        <p class="change_link">

          <a  id="button3" href="contacto.html"> Contacto </a>
        </p>
      </form>
  </div>

  <footer>
    <label id=pie>
    Trabajo de fin de grado de Nerea Rodera S&aacute;nchez  </label>
    <p> </p>
    <small id=cp>Todos los derechos reservados</small>
  </footer>

</body>
```

C.2. Contacto

```
<!DOCTYPE html>
<html lang="es">
<head>
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></
    script>
  <script src="javascript.js"language="JavaScript1.2" charset="utf-8"></script>

  <meta charset="iso-8559-1" />
  <meta http-equiv="content-type" content="text/html; charset=utf-8" />
  <meta name="descritpion" content="Control de riegos" />
  <meta name="keywords" content="riegos, ambientes, control" />

  <title>Control de riegos por ambientes</title>
  <link rel="stylesheet" href="mihojadeestilos_cont.css" />

</head>

<body>

<a id="escudo" target="_" href="http://www.uc3m.es">
    <img src="uc3m.png" alt="UC3M">
</a>
```

```
22
  <div id="wrapper">
24   <form method=post autocomplete="on">
       <h2>DATOS DE CONTACTO</h2>
26
         <label>
28         <table id="datos">
             <tr>
30             <td>Nombre</td>
               <td>Nerea</td>
32           </tr>
             <tr>
34           <td>Apellidos</td>
               <td>Rodera Sánchez</td>
36           </tr>
             <tr>
38             <td>Organización</td>
               <td>Universidad Carlos III de Madrid</td>
40           </tr>
             <tr>
42             <td>Correo de contacto</td>
               <td>100314974@alumnos.uc3m.es</td>
44           </tr>
             <tr>
46             <td>Título del proyecto</td>
               <td>Plataforma de riego controlado basado en microcontrolador de
    bajo coste y amplia conectividad</td>
48           </tr>
           </table>
50
         <a  id="button3" href="index.html"> Volver </a>
52     </label>
       </form>
54 </div>

56 <footer>
     <label id=pie>
58    Trabajo de fin de grado de Nerea Rodera S&aacute;nchez  </label>
     <p> </p>
60   <small id=cp>Todos los derechos reservados</small>
   </footer>
62 </body>

64 </html>
```

C.3. Unirse

```
<!DOCTYPE html>
2 <html lang="es">
<head>
4
  <script src="javascript.js"language="JavaScript1.2" charset="utf-8"></script>
6
```

```
  <meta charset="iso-8559-1" />
  <meta http-equiv="content-type" content="text/html; charset=utf-8" />
  <meta name="descritpion" content="Control de riegos" />
  <meta name="keywords" content="riegos, ambientes, control" />

  <title>Control de riegos por ambientes</title>
  <link rel="stylesheet" href="mihojadeestilos.css" />

</head>

<body>

<script>
function login(){

  var storedname = localStorage.getItem('usernamesignup');
    var storedpassword = localStorage.getItem('passwordsignup');

    var username = document.getElementById('username');
    var userpassword = document.getElementById('userpassword');

  if( username.value == storedname && userpassword.value == storedpassword) {
        window.location.href="pagina_principal_v1.html"
    }else {
        alert('Por favor, revise su usuario y contrase\u00f1a.');
    }
}
</script>

<a id="escudo" target="_" href="http://www.uc3m.es">
    <img src="uc3m.png" alt="UC3M">
</a>


  <div id="wrapper">
      <form method=post autocomplete="on">
        <h2>ACCEDER</h2>
        <p>
          <label for="username" class="uname"> Nombre de usuario </label>
          <input id="username" name="username" required="required" type="text"
    placeholder="Usuario" />
        </p>
        <p>
          <label for="userpassword" class="youpasswd"> Contrase&ntilde;a </label>
          <input id="userpassword" name="userpassword" required="required" type="
    password" placeholder="Password" />
        </p>
        <p>
            <a onclick="login();" class="button">Acceder </a>
        </p>
        <p class="change_link">
        &iquest;A&uacute;n no se ha registrado?
          <a id="button2" href="register.html" class="to_register">Unirse</a>
        </p>
      </form>
  </div>
  <footer>
```

```
62      <label id=pie>
        Trabajo de fin de grado de Nerea Rodera S&aacute;nchez  </label>
64      <!--<address id=mail>nrodera@gmail.com</address>-->
        <p> </p>
66      <small id=cp>Todos los derechos reservados</small>
    </footer>
68  </body>

70  </html>
```

C.4. Registro

```
<!DOCTYPE html>
2  <html lang="es">
   <head>
4
     <script src="javascript.js"language="JavaScript1.2" charset="utf-8"></script>
6
     <meta charset="iso-8559-1" />
8    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
     <meta name="descritpion" content="Control de riegos" />
10   <meta name="keywords" content="riegos, ambientes, control" />

12   <title>Control de riegos por ambientes</title>
     <link rel="stylesheet" href="mihojadeestilos_reg.css" />
14
   </head>
16
   <body>
18
   <script>
20 function check(){
   /**ALMACENAR DATOS: https://stackoverflow.com/questions/34299555/javascript-login-
       register-with-localstorage **/
22
   var usernamesignup = document.getElementById('usernamesignup');
24 var passwordsignup = document.getElementById('passwordsignup');

26 localStorage.setItem('usernamesignup', usernamesignup.value);
   localStorage.setItem('passwordsignup', passwordsignup.value);
28


30 /**COMPROBACIÓN DE USUARIO**/
   /*Es necesario hacer esto dado que no funciona PHP en el micro y el required con
       la
32 restricción de caracteres no funciona*/

34 var user=document.getElementById('usernamesignup').value;

36   if (user.length<1){
       window.alert("Introduzca un usuario correcto");
38   }
     else{
```

```
40
/**COMPROBACIÓN DE CONTRASEÑAS**/
42
  var pass1=document.getElementById('passwordsignup');
44  var pass2=document.getElementById('passwordsignup_confirm');
  //Para comprobar la longitud de la cadena se necesita que la var sea .value
46  var lon1=document.getElementById('passwordsignup').value;


48
  if (pass1.value==pass2.value){
50    if (lon1.length>4){
      if (lon1.length<13){
52        window.location.href="formulario_v1.html";
      }else if (lon1.length>12){
54        window.alert("La contrase\u00f1a sobrepasa los 12 caracteres permitidos");
      }
56    } else if (lon1.length<5){
      window.alert("Es necesario que su contrase\u00f1a tenga, al menos, 5
    caracteres");
58    }
  }else if (pass1.value!== pass2.value){
60      window.alert("Las contrase\u00f1as no concuerdan");
  }
62 }
 } //Fin función check,
64 </script>

66 <a id="escudo" target="_" href="http://www.uc3m.es">
              <img src="uc3m.png" alt="UC3M">
68 </a>

70  <div id="wrapper">
      <form  name="registro" method=get autocomplete="on">
72      <h2>REGISTRARSE</h2>
        <p>
74        <label for="usernamesignup" class="uname">Nombre de usuario</label>
          <input id="usernamesignup" name="usernamesignup" required="required"
    type="text" placeholder="User1234" />
76      </p>

78      <p>
          <label for="passwordsignup" class="youpasswd">Escriba su contrase&ntilde
    ;a, mínimo 5 caracteres y máximo 12. </label>
80        <input id="passwordsignup" name="passwordsignup" required="required"
    type="password" placeholder="Password" pattern="[A-Za-z0-9!?-]{5,12}"/>
      </p>
82      <p>
          <label for="passwordsignup_confirm" class="youpasswd">Por favor,
    confirme su contrase&ntilde;a </label>
84        <input id="passwordsignup_confirm" name="password_confirm" required="
    required" type="password" placeholder="Password" pattern="[A-Za-z0-9!?-]{5,12}
    "/>
      </p>
86      <p>
          <a  class="button" onclick="check();"> Siguiente</a>
88      </p>
        <p class="change_link">
```

```
90          &iquest;Ya se ha registrado?
              <a id="button2" href="login.html" class="to_register"> Acceder </a>
92
            </p>
94        </form>
    </div>
96
    <footer>
98    <label id=pie>
      Trabajo de fin de grado de Nerea Rodera S&aacute;nchez
100   <p> </p>
      <small id=cp>Todos los derechos reservados</small>
102 </footer>
</body>
104
</html>
```

C.5. Formulario

```
1 <!DOCTYPE html>
<html lang="es">
3 <head>

5   <script src="javascript.js"language="JavaScript1.2" charset="utf-8"></script>

7   <meta charset="iso-8559-1" />
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
9   <meta name="descritpion" content="Control de riegos" />
    <meta name="keywords" content="riegos, ambientes, control" />
11
    <title>Control de riegos por ambientes</title>
13  <link rel="stylesheet" href="mihojadeestilos_form.css" />


15
</head>
17
<body>
19 <script>
/****FORMULARIO****/
21
function check_form() {
23   var count=0;
     var count_terr=0;
25
     if (document.getElementById("checkbox1").checked){
27     count++;
     }
29   if (document.getElementById("checkbox2").checked){
       count++;
31   }
     if (document.getElementById("checkbox3").checked){
33     count++;
     }
```

```
35    if (document.getElementById("checkbox4").checked){
        count++;
37    }
      if (document.getElementById("checkbox5").checked){
39      count++;
      }
41    if (document.getElementById("checkbox6").checked){
        count++;
43    }
      if (document.getElementById("checkbox7").checked){
45      count++;
      }
47    if (document.getElementById("checkbox8").checked){
        count++;
49    }
      if (document.getElementById("checkbox9").checked){
51      count++;
      }
53    if(document.querySelector(".tipo_terr").value!==""){
      count_terr++;
55    }

57    if (count>1){
      window.alert("Por favor, elija sólo un ambiente");
59    }else if (count==0){
      window.alert("Elija al menos un ambiente");
61    }else{
        if (count_terr==0){
63      window.alert("Por favor, elija el tipo de terreno que desea");
        }else{
65      window.location.href="pagina_principal_v1.html"
        }
67    }


69
  }
71</script>

73<header id="cabecera">
  </header>
75

      <a id="escudo" target="_" href="http://www.uc3m.es">
77              <img src="uc3m.png" alt="UC3M">
          </a>
79

      <div id="wrapper">
81      <h2>Selecci&oacute;n de ambientes</h2>
        <p id="instr">Seleccione el tipo de planta que desea. En caso de no
      encontrarla, elija la que tenga las condiciones
83      de riego m&aacute;s parecidas.</p>
        <h3 id="ext">EXTERIOR</h3>
85        <div id="box1"><input id="checkbox1" type="checkbox" value="1" onclick="
      checked1();"/>&Aacute;rboles<br></div>
          <div id="box2"><input id="checkbox2" type="checkbox" value="1" onclick="
      checked2();"/>Palmeras<br></div>
87        <div id="box3"><input id="checkbox3" type="checkbox" value="1" onclick="
      checked3();"/>Huerto<br></div>
```

```
        <div id="box4"><input id="checkbox4" type="checkbox" value="1" onclick="
    checked4();" />Flores<br></div>
89      <div id="box5"><input id="checkbox5" type="checkbox" value="1" onclick="
    checked5();" />C&aacute;ctus<br></div>
        <div id="box6"><input id="checkbox6" type="checkbox" value="1" onclick="
    checked6();" />C&eacute;sped<br></div>
91
      <h3 id="int">INTERIOR</h3>
93
        <div id="box7"><input id="checkbox7" type="checkbox" value="1" onclick="
    checked7();" />Plantas con flor<br></div>
95      <div id="box8"><input id="checkbox8" type="checkbox" value="1" onclick="
    checked8();" />Plantas sin flor<br></div>
        <div id="box9"><input id="checkbox9" type="checkbox" value="1" onclick="
    checked9();" />Orqu&iacute;deas<br></div>
97
      <p id="instr2">Seleccione el tipo de terreno que mejor se adapte al suyo.</p
    >
99    <div id="tipo_terr">
      <select name="tipo_terr" class="tipo_terr">
101     <option id="val_null" value=""> </option>
        <option id="terr1" value ="arboles" onclick="selecc_terr1();">Terreno &
    aacute;rido</option>
103     <option id="terr2" value ="huerto" onclick="selecc_terr2();">Terreno h&
    uacute;medo</option>

105   </select>


107
  </div>
109
  <a id="button" onclick="check_form();"> Confirmar </a>
111
  <footer>
113 <label id=pie>
      Trabajo de fin de grado de Nerea Rodera S&aacute;nchez  </label>
115   <p> </p>
      <small id=cp>Todos los derechos reservados</small>
117 </footer>
</body>
119
</html>
```

C.6. Página principal

```
<!DOCTYPE html>
2 <html lang="es">
<head>
4
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"><
    /script>
6 <script src="javascript.js"language="JavaScript1.2" charset="utf-8"></script>
```

```
8   <meta charset="iso-8559-1" />
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
10  <meta name="descritpion" content="Control de riegos" />
    <meta name="keywords" content="riegos, ambientes, control" />
12
    <title>Control de riegos por ambientes</title>
14  <link rel="stylesheet" href="mihojadeestilos_ppal.css" />


16



18


20 </head>

22 <body>


24
<script> //http://www.htmlgoodies.com/beyond/javascript/article.php/3724571/Using-
    Multiple-JavaScript-Onload-Functions.htm
26


28 //Función para mostrar si el ambiente es exterior o interior
   function act_amb1(){
30   var req = false;

32    function act1Complete()
      {
34        if(req.readyState == 4)
          {
36            if(req.status == 200)
              {
38                document.getElementById("amb1state").innerHTML = "<div>" +
                                                    req.responseText + "</div>";
40            }
          }
42    }

44    if(window.XMLHttpRequest)
      {
46        req = new XMLHttpRequest();
      }
48    else if(window.ActiveXObject)
      {
50        req = new ActiveXObject("Microsoft.XMLHTTP");
      }
52    if(req)
      {
54        req.open("GET", "/cgi-bin/act_amb1?id" + Math.random(), true);
          req.onreadystatechange = act1Complete;
56        req.send(null);
      }
58  var t = setTimeout(act_amb1, 2000);
}
60
//Función para mostrar el tipo de ambiente
62 function display_amb1(){
     var req = false;
```

```
64
    function dis1Complete()
66  {
        if(req.readyState == 4)
68      {
            if(req.status == 200)
70          {
                document.getElementById("amb1tipo").innerHTML = "<div>" +
72                                                  req.responseText + "</div>";
            }
74      }
    }
76
    if(window.XMLHttpRequest)
78  {
        req = new XMLHttpRequest();
80  }
    else if(window.ActiveXObject)
82  {
        req = new ActiveXObject("Microsoft.XMLHTTP");
84  }
    if(req)
86  {
        req.open("GET", "/cgi-bin/display_amb1?id" + Math.random(), true);
88      req.onreadystatechange = dis1Complete;
        req.send(null);
90  }
  var t = setTimeout(display_amb1, 2000);
92}

94//Función para mostrar el tiempo de regar
function display_prox1(){
96   var req = false;

98    function prox1Complete()
    {
100     if(req.readyState == 4)
        {
102         if(req.status == 200)
            {
104             document.getElementById("amb1prox").innerHTML = "<div>" +
                                                req.responseText + "</div>";
106         }
        }
108 }

110 if(window.XMLHttpRequest)
    {
112     req = new XMLHttpRequest();
    }
114 else if(window.ActiveXObject)
    {
116     req = new ActiveXObject("Microsoft.XMLHTTP");
    }
118 if(req)
    {
120     req.open("GET", "/cgi-bin/display_prox1?id" + Math.random(), true);
```

```
        req.onreadystatechange = prox1Complete;
122         req.send(null);
    }
124   var t = setTimeout(display_prox1, 2000);
}
126
//Función para mandar el tipo de terreno
128 function display_terr(){
   var req = false;
130
    function terrComplete()
132 {
        if(req.readyState == 4)
134     {
            if(req.status == 200)
136         {
                document.getElementById("tipo_terr").innerHTML = "<div>" +
138                                                     req.responseText + "</div>";
            }
140     }
    }
142
    if(window.XMLHttpRequest)
144 {
        req = new XMLHttpRequest();
146 }
    else if(window.ActiveXObject)
148 {
        req = new ActiveXObject("Microsoft.XMLHTTP");
150 }
    if(req)
152 {
        req.open("GET", "/cgi-bin/display_terr?id" + Math.random(), true);
154     req.onreadystatechange = terrComplete;
        req.send(null);
156 }
  var t = setTimeout(display_terr, 2000);
158 }
function name_display(){
160
     var storedname = localStorage.getItem('usernamesignup');
162  document.getElementById("username_log").innerHTML= 'Bienvenido, ' +
   storedname;
}
164
//Mandar orden de leer sensores
166 function mostrar()
{
168 var req = false;

170 function mostrarComplete()
    {
172     if(req.readyState == 4)
        {
174         if(req.status == 200)
            {
176             document.getElementById("amb1temp").innerHTML = "<div>" +
```

```
                                                    req.responseText + "</div>";
178                 }
            }
180     }

182     if(window.XMLHttpRequest)
        {
184         req = new XMLHttpRequest();
        }
186     else if(window.ActiveXObject)
        {
188         req = new ActiveXObject("Microsoft.XMLHTTP");
        }
190     if(req)
        {
192         req.open("GET", "/cgi-bin/mostrar?id" + Math.random(), true);
            req.onreadystatechange = mostrarComplete;
194         req.send(null);
        }
196   var t = setTimeout(mostrar, 2000);
}
198
//Funciones para leer y mostrar la hora.
200 function startTime() {
    var today = new Date();
202   var h = today.getHours();
    var m = today.getMinutes();
204   var s = today.getSeconds();
    m = checkTime(m);
206   s = checkTime(s);
    document.getElementById('hour').innerHTML = h + ":" + m + ":" + s;
208   var t = setTimeout(startTime, 500);
}
210 function checkTime(i) {
    if (i < 10) {i = "0" + i};
212   return i;
}
214 function startDate(){
    var today= new Date();
216   var d = today.getDate();
    var mt = today.getMonth();
218   var y = today.getFullYear();
    mt = mt + 1;
220   document.getElementById('date').innerHTML = d + "/" + mt + "/" + y;
    var t = setTimeout(startTime, 500);
222


224 }
//funcion comunicar web con micro
226 function amb1riego(){
    var req = false;
228
    function riego1Complete()
230   {
      if(req.readyState == 4)
232     {
        if(req.status == 200)
```

```
234       {
            document.getElementById("amb1riego").innerHTML = "<div>" +
236                         req.responseText + "</div>";
          }
238     }
    }
240
    if(window.XMLHttpRequest)
242 {
      req = new XMLHttpRequest();
244 }
    else if(window.ActiveXObject)
246 {
      req = new ActiveXObject("Microsoft.XMLHTTP");
248 }

250 if(req)
    {
252   req.open("GET", "/cgi-bin/amb1riego?id" + Math.random(), true);
      req.onreadystatechange = riego1Complete;
254   req.send(null);
    }
256


258 }

260 function advert(){
window.alert("Por favor, espere a que se carguen los datos de la página. Esto
    puede tardar algunos segundos...");
262 }

264 function addLoadEvent(func) {
    var oldonload = window.onload;
266 if (typeof window.onload != 'function') {
      window.onload = func;
268 } else {
      window.onload = function() {
270     if (oldonload) {
          oldonload();
272     }
        func();
274   }
    }
276 }

278 addLoadEvent(startTime);
    addLoadEvent(startDate);
280 addLoadEvent(name_display);

282 addLoadEvent(display_terr);
    addLoadEvent(mostrar);
284
    addLoadEvent(act_amb1);
286 addLoadEvent(display_amb1);
    addLoadEvent(estat_1);
288 addLoadEvent(display_prox1);
```

```
290 addLoadEvent(temp_1);
    addLoadEvent(hum_1);
292 addLoadEvent(sens_1);
    addLoadEvent(advert);
294

296 </script>

298


300   <a id="escudo" target="_" href="http://www.uc3m.es">
        <img src="uc3m_large.png" alt="UC3M">
302   </a>

304   <div id="wrapper">
      <div id="hour"> </div>
306   <div id="date"> </div>

308   <img id ="logo" src="logo.png" alt="USER">
      <div id="username_log"></div>
310
      <a id="redef" onclick="reset_amb();" href="index.html">Cerrar sesi&oacute;n</a>
312
        <div id="cuadro1">
314       <div id="intro"> Usted ha elegido un ambiente de:</div>
          <div id="amb1state"> </div>
316       <div id="intro2"> Su ambiente:</div>
          <div id="amb1tipo"> </div>
318       <div id="intro3"> El terreno es:</div>
          <div id="tipo_terr"> </div>
320       <div id="DATOS1">Información del ambiente</div>
            <div id="amb1temperatura">Temperatura:</div>
322         <div id="amb1temp">-</div>
            <div id="amb1humedad">Humedad:</div>
324         <div id="amb1hum">-</div>
            <div id="amb1TA">Sensaci&oacute;n t&eacute;rmica:</div>
326         <div id="amb1ta">-</div>
          <div id="amb1riego_state">Estado del riego:</div>
328       <div id="amb1riego"> </div>
          <div id="amb1proximo">Pr&oacute;ximo riego:</div>
330       <div id="amb1prox">-</div>
        </div>
332


334
      <footer id=pie>
336     Trabajo de fin de grado de Nerea Rodera S&aacute;nchez
        <p> </p>
338     <small id=cp>Todos los derechos reservados</small>
      </footer>
340 </body>

342


344 </html>
```

Nota: En la parte de código correspondiente a JavaScript se encuentra la función *addLoadEvent*, adquirida en el link junto a la función. Esta sirve para, tras cargar la página, cargar las funciones en el orden que aparecen. Para no sobrecargar el código, muchas de las funciones que aquí se muestran, y que sirven para conectar el microprocesador con la web, no se han incluido; dejando tan sólo la función *amb1riego()* como ejemplo.

Anexos D

Anexo IV: Código C, programa de riego

En el siguiente anexo se presenta el código correspondiente a la realización del programa de riego, e irá incluido dentro del programa *enet_io*. Al igual que en el capítulo 6, se ha dividido en bloques para una mejor comprensión.

El objetivo de este anexo, como ya se mencionó en el anterior, es proporcionar el código en caso de que una persona externa desee hacer mejoras o modificaciones; o simplemente pueda servir de ayuda a la comprensión.

Todo el código aquí escrito ha sido desarrollado desde cero sin basarse en ningún programa o ejemplo previo.

D.1. Temporización

```
if(measure_done==1 && medir_tiempo==1){
        if(a>=300){
            a=0;
            minutos++;
        }
    }

      if (minutos>=60){
            minutos=0;
            horas++;
            horas_alm++;
      }
```

D.2. Establecer estación del año

```
if (horas_alm>=8){
            horas_alm=0;
```

```
            temp_alm[i]=Temp_IntegerPart;
            i++;
            if (i==7){
                i=0;
                for(j=0;j<7;j++){
                    if(temp_alm[j]<20){
                        cuenta_temp_cold++;
                    }
                }
                for(j=0;j<7;j++){
                    if(temp_alm[j]>35){
                        cuenta_temp_hot++;
                    }

                }
            }
    }

        if(cuenta_temp_hot>=2||cuenta_temp_cold<=2)
          {
            verano=1;
            invierno=0;
          }
          else
          {
            invierno=1;
            verano=0;
          }
```

D.3. Establecer tiempo de riego

Nota: Este código puede ser diferente del diagrama de flujo correspondiente en el capítulo 6 (ver figura 6.4).Esto se debe a que ha debido ser modificado para la simulación del programa, donde se riega el ambiente césped por minutos, y no por horas.

```
if (tipo1==1)
                {
                    if (verano==1){
                        if (terreno1==1){
                            tiempo_riego=6;
                        }
                        else{
                            tiempo_riego=12;
                        }
                        if (horas>=tiempo_riego){
                            ena_riego=1;
                            horas=0;
                            minutos=0;
                            mostrar_min=0;
                            a=0;
                            min_hasta_riego=0;
                            horas_hasta_riego=0;
```

```
                    }
                }
                else
                {
                    if (terreno1==1){
                        tiempo_riego=12;
                    }
                    else{
                        tiempo_riego=24;
                    }
                    if (horas>=tiempo_riego){
                        ena_riego=1;
                        horas=0;
                        minutos=0;
                        mostrar_min=0;
                        a=0;
                        min_hasta_riego=0;
                        horas_hasta_riego=0;
                    }
                }
            }
            else if(tipo2==1)
            {
                if (verano==1){
                    if (terreno1==1){
                        tiempo_riego=36;
                    }
                    else{
                        tiempo_riego=48;
                    }
                    if (horas>=tiempo_riego){
                        ena_riego=1;
                        horas=0;
                        minutos=0;
                        mostrar_min=0;
                        a=0;
                        min_hasta_riego=0;
                        horas_hasta_riego=0;
                    }
                }
                else
                {
                    if (terreno1==1){
                        tiempo_riego=72;
                    }
                    else{
                        tiempo_riego=96;
                    }
                    if (horas>=tiempo_riego){
                        ena_riego=1;
                        horas=0;
                        minutos=0;
                        mostrar_min=0;
                        a=0;
                        min_hasta_riego=0;
                        horas_hasta_riego=0;
                    }
```

```
75                      }
                    }
77                  else if(tipo3==1)
                    {
79                      if (verano==1){
                            if (terreno1==1){
81                              tiempo_riego=12;
                            }
83                          else{
                                tiempo_riego=24;
85                          }
                            if (horas>=tiempo_riego){
87                              ena_riego=1;
                                horas=0;
89                              minutos=0;
                                mostrar_min=0;
91                              a=0;
                                min_hasta_riego=0;
93                              horas_hasta_riego=0;
                            }
95                      }
                        else
97                      {
                            if (terreno1==1){
99                              tiempo_riego=24;
                            }
101                         else{
                                tiempo_riego=48;
103                         }
                            if (horas>=tiempo_riego){
105                             ena_riego=1;
                                horas=0;
107                             minutos=0;
                                mostrar_min=0;
109                             a=0;
                                min_hasta_riego=0;
111                             horas_hasta_riego=0;
                            }
113                     }
                    }
115                 else if(tipo4==1)
                    {
117                     if (verano==1){
                            if (terreno1==1){
119                             tiempo_riego=24;
                            }
121                         else{
                                tiempo_riego=48;
123                         }
                            if (horas>=tiempo_riego){
125                             ena_riego=1;
                                horas=0;
127                             minutos=0;
                                mostrar_min=0;
129                             a=0;
                                min_hasta_riego=0;
131                             horas_hasta_riego=0;
```

```
                    }
                }
                else
                {
                    if (terreno1==1){
                        tiempo_riego=48;
                    }
                    else{
                        tiempo_riego=96;
                    }
                    if (horas>=tiempo_riego){
                        ena_riego=1;
                        horas=0;
                        minutos=0;
                        mostrar_min=0;
                        a=0;
                        min_hasta_riego=0;
                        horas_hasta_riego=0;
                    }
                }
            }
            else if(tipo5==1)
            {
                if (verano==1){
                    if (terreno1==1){
                        tiempo_riego=120;
                    }
                    else{
                        tiempo_riego=168;
                    }
                    if (horas>=tiempo_riego){
                        ena_riego=1;
                        horas=0;
                        minutos=0;
                        mostrar_min=0;
                        a=0;
                        min_hasta_riego=0;
                        horas_hasta_riego=0;
                    }
                }
                else
                {
                    if (terreno1==1){
                        tiempo_riego=240;
                    }
                    else{
                        tiempo_riego=336;
                    }
                    if (horas>=tiempo_riego){
                        ena_riego=1;
                        horas=0;
                        minutos=0;
                        mostrar_min=0;
                        a=0;
                        min_hasta_riego=0;
                        horas_hasta_riego=0;
                    }
```

```
189                     }
                    }
191                 else if(tipo6==1)
                    {
193                     if (verano==1){
                            if (terreno1==1){
195                             tiempo_riego=3;
                            }
197                         else{
                                tiempo_riego=5;
199                         }
                            tiempo_riego_min=1;//En simulación, para saber que estamos
    midiendo minutos
201                         if (minutos>=tiempo_riego){
                                ena_riego=1;
203                             horas=0;
                                minutos=0;
205                             mostrar_min=0;
                                a=0;
207                             min_hasta_riego=0;
                                horas_hasta_riego=0;
209                         }
                        }
211                     else
                        {
213                         if (terreno1==1){
                                tiempo_riego=6;
215                         }
                            else{
217                             tiempo_riego=10;
                            }
219                         tiempo_riego_min=1;
                            if (horas>=tiempo_riego){
221                             ena_riego=1;
                                horas=0;
223                             minutos=0;
                                mostrar_min=0;
225                             a=0;
                                min_hasta_riego=0;
227                             horas_hasta_riego=0;
                            }
229                     }
                    }
231                 else if(tipo7==1)
                    {
233                     if (verano==1){
                            if (terreno1==1){
235                             tiempo_riego=120;
                            }
237                         else{
                                tiempo_riego=240;
239                         }
                            if (horas>=tiempo_riego){
241                             ena_riego=1;
                                horas=0;
243                             minutos=0;
                                mostrar_min=0;
```

```
245                         a=0;
                            min_hasta_riego=0;
247                         horas_hasta_riego=0;
                        }
249                 }
                    else
251                 {
                        if (terreno1==1){
253                         tiempo_riego=240;
                        }
255                     else{
                            tiempo_riego=480;
257                     }
                        if (horas>=tiempo_riego){
259                         ena_riego=1;
                            horas=0;
261                         minutos=0;
                            mostrar_min=0;
263                         a=0;
                            min_hasta_riego=0;
265                         horas_hasta_riego=0;
                        }
267                 }
            }
269         else if(tipo8==1)
            {
271             if (verano==1){
                    if (terreno1==1){
273                     tiempo_riego=168;
                    }
275                 else{
                        tiempo_riego=336;
277                 }
                    if (horas>=tiempo_riego){
279                     ena_riego=1;
                        horas=0;
281                     minutos=0;
                        mostrar_min=0;
283                     a=0;
                        min_hasta_riego=0;
285                     horas_hasta_riego=0;
                    }
287             }
                else
289             {
                    if (terreno1==1){
291                     tiempo_riego=336;
                    }
293                 else{
                        tiempo_riego=672;
295                 }
                    if (horas>=tiempo_riego){
297                     ena_riego=1;
                        horas=0;
299                     minutos=0;
                        mostrar_min=0;
301                     a=0;
```

```
                    min_hasta_riego=0;
                    horas_hasta_riego=0;
                }
            }
        }
        else if(tipo9==1)
        {
            if (verano==1){
                if (terreno1==1){
                    tiempo_riego=48;
                }
                else{
                    tiempo_riego=96;
                }
                if (horas>=tiempo_riego){
                    ena_riego=1;
                    horas=0;
                    minutos=0;
                    mostrar_min=0;
                    a=0;
                    min_hasta_riego=0;
                    horas_hasta_riego=0;
                }
            }
            else
            {
                if (terreno1==1){
                    tiempo_riego=96;
                }
                else{
                    tiempo_riego=192;
                }
                if (horas>=tiempo_riego){
                    ena_riego=1;
                    horas=0;
                    minutos=0;
                    mostrar_min=0;
                    a=0;
                    min_hasta_riego=0;
                    horas_hasta_riego=0;
                }
            }
        }
```

D.4. Mostrar tiempo hasta próximo riego

```
horas_hasta_riego=tiempo_riego-horas;

  if (horas_hasta_riego==1){
    min_hasta_riego=60-minutos;
    mostrar_min=1;
  }
```

```
  if (tiempo_riego_min==1){
      min_hasta_riego=tiempo_riego-minutos;
    mostrar_min=1;
  }
```

D.5. Cálculo de la temperatura aparente

```
 if (measure_done==1){
            AT_Float= -9.93122+(1.186145*Temp_Float)+(0.122310*Hum_Float);
            AT_IntegerPart = (int32_t) AT_Float;
            AT_FractionPart = (int32_t) (AT_Float * 1000.0f);
        }
```

D.6. Comprobar si los parámetros son adecuados

```
if (ena_riego==1 && estado==0){
            if (Temp_IntegerPart>40)
            {

                estado=1;
                medir_tiempo=0;
            }
            else if (Temp_IntegerPart<0)
            {

                estado=2;
                medir_tiempo=0;
            }
            else{
                act_riego=1;
            }
        }
```

D.7. Simulación de lluvia

```
if (Hum_IntegerPart>50)
        {
            estado=3;
            horas=0;
            minutos=0;
            a=0;
            mostrar_min=0;
            min_hasta_riego=0;
            horas_hasta_riego=0;
```

```
        }
```

D.8. Actualizar estado

```
if (estado==1){
            if (Temp_IntegerPart<40)
            {
                estado=0;
                act_riego=1;
                a=0;
                minutos=0;
                horas=0;
                medir_tiempo=1;
            }
        }
        else if (estado==2){
            if (Temp_IntegerPart>0)
            {
                estado=0;
                act_riego=1;
                a=0;
                minutos=0;
                horas=0;
                medir_tiempo=1;
            }
        }
        else if (estado==3){
            if(Hum_IntegerPart<50)
            {
                estado=0;
            }
        }
```

D.9. Regar

```
if (act_riego==1){
        //Activar Pin que ponga a 1 la salida
            if(tipo6==1){
                GPIOPinWrite(GPIO_PORTL_BASE, GPIO_PIN_1, GPIO_PIN_1);      //GPIO
    PL1
            }
            else{
                GPIOPinWrite(GPIO_PORTL_BASE, GPIO_PIN_0, GPIO_PIN_0);
            }
            estado=4;
       ena_riego=0;
       act_riego=1;
            if (a_riego>=300){
                min_riego++;
```

```
14                  a_riego=0;}
                    if (tipo1==1)
16                  {
                        if (min_riego>=5)
18                      {
                            act_riego=0;
20                          min_riego=0;
                            riego1=0;
22                          estado=0;
                            GPIOPinWrite(GPIO_PORTL_BASE, GPIO_PIN_0, 0x0);
24                      }
                    }
26                  else if(tipo2==1)
                    {
28                      if (min_riego>=3)
                        {
30                          act_riego=0;
                            min_riego=0;
32                          riego1=0;
                            estado=0;
34                          GPIOPinWrite(GPIO_PORTL_BASE, GPIO_PIN_0, 0x0);
                        }
36                  }
                    else if(tipo3==1)
38                  {
                        if (min_riego>=5)
40                      {
                            act_riego=0;
42                          min_riego=0;
                            riego1=0;
44                          estado=0;
                            GPIOPinWrite(GPIO_PORTL_BASE, GPIO_PIN_0, 0x0);
46                      }
                    }
48                  else if(tipo4==1)
                    {
50                       if (min_riego>=2)
                         {
52                           act_riego=0;
                             min_riego=0;
54                           riego1=0;
                             estado=0;
56                           GPIOPinWrite(GPIO_PORTL_BASE, GPIO_PIN_0, 0x0);
                         }
58                  }
                    else if(tipo5==1)
60                  {
                       if (min_riego>=2)
62                     {
                           act_riego=0;
64                         min_riego=0;
                           riego1=0;
66                         estado=0;
                           GPIOPinWrite(GPIO_PORTL_BASE, GPIO_PIN_0, 0x0);
68                     }
                    }
70                  else if(tipo6==1)
```

```
                {
72                  if (min_riego>=1)
                    {
74                      act_riego=0;
                        min_riego=0;
76                      riego1=0;
                        estado=0;
78                      GPIOPinWrite(GPIO_PORTL_BASE, GPIO_PIN_1, 0x0);
                    }
80              }
                else if(tipo7==1)
82              {
                    if (min_riego>=3)
84                  {
                        act_riego=0;
86                      min_riego=0;
                        riego1=0;
88                      estado=0;
                        GPIOPinWrite(GPIO_PORTL_BASE, GPIO_PIN_0, 0x0);
90                  }
                }
92              else if(tipo8==1)
               {
94                  if (min_riego>=3)
                    {
96                      act_riego=0;
                        min_riego=0;
98                      riego1=0;
                        estado=0;
100                     GPIOPinWrite(GPIO_PORTL_BASE, GPIO_PIN_0, 0x0);
                    }
102             }
                else if(tipo9==1)
104             {
                   if (min_riego>=2)
106                {
                       act_riego=0;
108                    min_riego=0;
                       riego1=0;
110                    estado=0;
                       GPIOPinWrite(GPIO_PORTL_BASE, GPIO_PIN_0, 0x0);
112
                   }
114            }
        }
```

Bibiliografía

[1] Bombas HASA. Accesorios para bombas de aguas limpias. https://www.bombashasa.com/es/accesorios-bombas-aguas-limpias/. Consultada el 22 de Agosto de 2017.

[2] Bosh Sensortec. Bmp180. https://www.bosch-sensortec.com/bst/products/all_products/bmp180. Consultada el 6 de Agosto de 2017.

[3] Certicalia. Normativa y requisitos del análisis de agua para riego. https://www.certicalia.com/analisis-de-agua-para-riego/normativa-y-requisitos-del-analisis-de-agua-para-riego. Consultada el 30 de Julio de 2017.

[4] Disumtec. Electrovalvula 4/3 centro cerrado e6. https://www.disumtec.com/ELECTROVALVULA-4/3-CENTRO-CERRADO-E-6. Consultada el 15 de Septiembre de 2017.

[5] DIY START. Irrigation system - past and present. https://www.diystart.com/irrigation/irrigation-system-past-and-present, 2015. Consultada el 10 de Agosto de 2017.

[6] Salvat Editores. *Historia universal: El origen*. Salvat, Madrid, ES, 2004.

[7] Electrónica embajadores. Electrónica embajadores. https://www.electronicaembajadores.com/es/Productos/Detalle/CJPPR03004/cajas-para-electronica/cajas-convencionales-de-plastico/caja-convencional-plastico-retex-155x95x57mm-serie-103-33103004. Consultada el 15 de Septiembre de 2017.

[8] Enrique Vicente Bonet Esteban. Lenguaje c. informatica.uv.es/estguia/ATD/apuntes/laboratorio/Lenguaje-C.pdf. Consultada el 3 de Agosto de 2017.

[9] ETESA. Sensación térmica. http://www.hidromet.com.pa/sensacion_termica.php, 2009. Consultada el 18 de Agosto de 2017.

[10] Exosite. Exosite. https://exosite.com/platform/. Consultada el 5 de Julio de 2017.

[11] Fliwer. Conoce al guardián de tus plantas. http://www.fliwer.com/#. Consultada el 27 de Julio de 2017.

[12] Juan Diego Gauchat. *El gran libro de HTML5, CSS3 Y Javascript*. Marcombo, S.A, Barcelona, ES, 2012.

[13] Carmela Grandes. *Las plantas que tenemos en casa*. GRAFALCO, S.A, Madrid, ES, 1998.

[14] HTML Goodies. Using multiple javascript onload functions. http://www.htmlgoodies.com/beyond/javascript/article.php/3724571/Using-Multiple-JavaScript-Onload-Functions.htm. Consultada el 20 de Julio de 2017.

[15] Iberdrola. Riego inteligente. https://www.iberdrola.es/hogar/servicios/equipos/riego-inteligente#. Consultada el 26 de Julio de 2017.

[16] Idis. Intelliwater. http://www.idiscompany.com/. Consultada el 26 de Julio de 2017.

[17] Intersil. Isl29023. http://www.intersil.com/en/products/optoelectronics/ambient-light-sensors/light-to-digital-sensors/ISL29023.html. Consultada el 6 de Agosto de 2017.

[18] Leroy Merlin. Aspersores. http://www.leroymerlin.es/productos/jardin/riego/aspersores.html. Consultada el 5 de Septiembre de 2017.

[19] Leroy Merlin. Bombas de agua. http://www.leroymerlin.es/productos/jardin/bombas_de_agua.html. Consultada el 20 de Agosto de 2017.

[20] Leroy Merlin. La inteligencia artificial llevada al riego de tu jardín de la mano de fliwer. http://www.leroymerlin.es/productos/jardin/riego/programadores_de_riego/fliwer.html. Consultada el 27 de Julio de 2017.

[21] MDN, Mozilla Developer Network. Css. https://developer.mozilla.org/es/docs/Web/CSS, 2016. Consultada el 5 de Agosto de 2017.

[22] MDN, Mozilla Developer Network. Css3. https://developer.mozilla.org/es/docs/Web/CSS/CSS3, 2016. Consultada el 5 de Agosto de 2017.

[23] MDN, Mozilla Developer Network. Html5. https://developer.mozilla.org/es/docs/HTML/HTML5, 2017. Consultada el 5 de Agosto de 2017.

[24] MDN, Mozilla Developer Network. Javascript. https://developer.mozilla.org/es/docs/Web/CSS/CSS3, 2017. Consultada el 7 de Agosto de 2017.

[25] Microsoft. ¿qué es dhcp? https://technet.microsoft.com/es-es/library/dd145320(v=ws.10).aspx. Consultada el 14 de Agosto de 2017.

[26] Monsolar. Kit solar salida a 12 voltios y 400wh/día consumo vacacional - b1. https://www.monsolar.com/kit-solar-basico-iluminacion-salida-a-12v-y-consumo-muy-bajo.html. Consultada el 2 de Septiembre de 2017.

[27] Monsolar. Regulador solar epsolar ls1024b 12/24v y 10a. https://www.monsolar.com/regulador-epsolar-ls1024-12-24v-y-10a.html. Consultada el 2 de Septiembre de 2017.

[28] Notepad ++. Notepad ++. https://notepad-plus-plus.org/. Consultada el 6 de Agosto de 2017.

[29] Novedades agrícolas. Bomba de riego. http://www.novedades-agricolas.com/es/riego/materiales-de-riego/bombas-de-riego. Consultada el 20 de Agosto de 2017.

[30] Novedades Agrícolas. Riego por goteo. http://www.novedades-agricolas.com/es/riego/sistemas-de-riego/riego-por-goteo, 2016. Consultada el 5 de Septiembre de 2017.

[31] Putty. Download putty. http://www.putty.org/. Consultada el 10 de Marzo de 2017.

[32] Quo, Francisco Cañizares. ¿quién inventó la manguera? http://www.quo.es/ser-humano/quien-invento-la-manguera, 2014. Consultada el 10 de Agosto de 2017.

[33] Riegos Grandal. Historia en imágenes del riego por aspersión 1872 - 1986. https://www.riegosgrandal.com/2014/07/24/historia-en-imagenes-del-riego-por-aspersion-1872-1986/, 2014. Consultada el 10 de Agosto de 2017.

[34] Sensirion. Digital humidity sensor sht2x (rh/t). https://www.sensirion.com/en/environmental-sensors/humidity-sensors/humidity-temperature-sensor-sht2x-digital-i2c-accurate/. Consultada el 6 de Agosto de 2017.

[35] Smart BioSystem. Smart biosystem. https://smartbiosystem.com/. Consultada el 27 de Julio de 2017.

[36] StackOverflow. Javascript: Login/register with localstorage. https://stackoverflow.com/questions/34299555/javascript-login-register-with-localstorage, 2015. Consultada el 3 de Junio de 2017.

[37] TDK. Mpu 9158. https://www.invensense.com/products/motion-tracking/9-axis/mpu-9150/. Consultada el 6 de Agosto de 2017.

[38] Texas Instruments. Sensor hub boosterpack. http://www.ti.com/tool/BOOSTXL-SENSHUB. Consultada el 23 de Febrero de 2017.

[39] Texas Instruments. *Meet the Tiva C Series TM4C1294*. Texas Instruments Incorporated, 2014.

[40] Texas Intrstruments. Code composer studio. http://www.ti.com/tool/ccstudio. Consultada el 6 de Agosto de 2017.

[41] Texas Intrstruments. Tmp006. http://www.ti.com/product/tmp006/. Consultada el 6 de Agosto de 2017.

[42] The IEEE and The Open Group. stdbool.h - boolean type and values. http://pubs.opengroup.org/onlinepubs/009695399/basedefs/stdbool.h.htmll, 2004. Consultada el 3 de Agosto de 2017.

[43] The IEEE and The Open Group. stdint.h - integer types. http://pubs.opengroup.org/onlinepubs/009695399/basedefs/stdint.h.html, 2004. Consultada el 3 de Agosto de 2017.

[44] The IEEE and The Open Group. string.h - string operations. http://pubs.opengroup.org/onlinepubs/7908799/xsh/string.h.html, 2004. Consultada el 3 de Agosto de 2017.

[45] Tiva. Clp unbox. http://processors.wiki.ti.com/index.php/CLP_Unbox, 2014. Consultada el 4 de Agosto de 2017.

[46] Traxco. Historia ancestral del riego agrícola. https://www.traxco.es/blog/noticias-agricolas/historia-ancestral-del-riego, 2010. Consultada el 10 de Agosto de 2017.

[47] W3 Schools. Css how to... https://www.w3schools.com/css/css_howto.asp. Consultada el 5 de Agosto de 2017.

[48] W3 Schools. Css syntax and selectors. https://www.w3schools.com/css/css_syntax.asp. Consultada el 5 de Agosto de 2017.

[49] W3 Schools. Display clock example. https://www.w3schools.com/js/tryit.asp?filename=tryjs_timing_clock. Consultada el 03 de Junio de 2017.

[50] W3 Schools. Html attributes. https://www.w3schools.com/tags/ref_attributes.asp. Consultada el 6 de Agosto de 2017.

[51] W3 Schools. Html class attribute. https://www.w3schools.com/tags/att_class.asp. Consultada el 5 de Agosto de 2017.

[52] W3 Schools. Html dom events. https://www.w3schools.com/jsref/dom_obj_event.asp. Consultada el 6 de Agosto de 2017.

[53] W3 Schools. Html dom queryselector() method. https://www.w3schools.com/jsref/met_document_queryselector.asp. Consultada el 12 de Agosto de 2017.

[54] W3 Schools. Html id attribute. https://www.w3schools.com/Tags/att_global_id.asp. Consultada el 5 de Agosto de 2017.

[55] W3 Schools. Html introduction. https://www.w3schools.com/html/html_intro.asp. Consultada el 5 de Agosto de 2017.

[56] W3 Schools. Javascript dates. https://www.w3schools.com/js/js_dates.asp. Consultada el 03 de Junio de 2017.

[57] W3 Schools. Window settimeout() method. https://www.w3schools.com/jsreF/met_win_settimeout.asp. Consultada el 14 de Junio de 2017.

[58] Wikipedia. Server side includes. https://es.wikipedia.org/wiki/Server_Side_Includes. Consultada el 5 de Julio de 2017.

[59] Wikipedia. Grados de protección ip. https://es.wikipedia.org/wiki/Grado_de_protecci%C3%B3n_IP, 2017. Consultada el 1 de Septiembre de 2017.

[60] Kepa Zubiri. *1.000 trucos de jardinería y plantas.* Servilibro, S.A, Madrid, ES.